THE JOHN OLIGER COMPANY

2068 FLOPPY DISK INTERFACE

FEATURING JLO SAFE V2

USER MANUAL

FILE ZERO MENUE/LOADER BY ROELOF MULDER
MODIFIED BY J. OLIGER

```
  1 CLEAR : LET M=VAL "16": DIM C$(VAL "178",VAL "2Ø"): LET S=S
GN PI: LET o=NOT PI: INK S: PAPER o: BORDER o: CLS
 1Ø LET Q$=" STEP "+CHR$ 1Ø+CHR$ o+"*K\"+CHR$ 17+CHR$ 14+CHR$ o
+CHR$ 25+CHR$ 235+"! &"+CHR$ S+" CLOSE #"+CHR$ 13+CHR$ 237+"VAL
!"+CHR$ M+"&"+CHR$ 14+CHR$ M+CHR$ 237+"VAL "+CHR$ 17+CHR$ 2Ø+CHR
$ o+">"+CHR$ 128+"PEEK ("+CHR$ 4+CHR$ 3+CHR$ 25+"Ø"+CHR$ 249+" O
UT "
 2Ø DIM F$(6,5): LET F$(S)="BASIC": LET F$(2)="N ARR": LET F$(3
)="C ARR": LET F$(4)="BYTES": LET F$(5)="STATE": LET F$(6)="VRBL
S"
 1ØØ DRAW 255,o: DRAW o,175: DRAW -255,o: DRAW o,-175: INK 5
 2ØØ LET A=(PEEK 23627+PEEK 23628*256+3589): LET C=INT (A/256):
POKE 23549,195: POKE 2355Ø,A-(C*256): POKE 23551,C: LET FI=USR 2
3549: LET Y=2: LET X=7: LET N$=C$(178, TO 16)
 3ØØ FOR /M: IF N$(LEN N$)=" " THEN LET N$=N$( TO LEN N$-S): NEX
T
 32Ø PRINT AT o,16-(LEN N$/2); OVER S;N$;AT o,8; OVER S;"(16 UNDER-
LINE CHARS)": INK 7
 4Ø5 LET F=S: LET C=INT (FI/18): LET DIF=INT ((FI/18-C)*18+.4):
LET N=17
 41Ø LET IT=S: IF N>=FI THEN LET N=FI: GO TO VAL "425"
 415 FOR I=S TO C: FOR /Y TO N+Y: PRINT AT M,8;C$(IT, TO 1Ø);" "
;F$(CODE C$(IT,11)+1): LET IT=IT+1: NEXT : LET M=N+S: GO SUB VAL
 "5ØØ": NEXT I: FOR /Y TO 19: PRINT AT M,8;"  (16 SPACES)      ": N
EXT : IF NOT DIF THEN GO TO 41Ø
 425 IF F THEN FOR /Y TO DIF-S+Y: PRINT AT M,8;C$(IT, TO 1Ø);" "
;F$(CODE C$(IT,11)+1): LET IT=IT+1: NEXT : LET M=DIF: IF N>=FI T
HEN LET F=o
 427 IF NOT F THEN LET IT=FI+S
 43Ø GO SUB VAL "5ØØ": GO TO VAL "41Ø"
 5ØØ LET L=M: FOR /o TO L-S: PRINT AT Y+M,X; INVERSE S;">": IF Q
$<>"" THEN PAUSE X: LET Q$="": PAUSE X
 51Ø LET A$=INKEY$: IF A$="" THEN LET Q$="p": GO TO VAL "51Ø"
 52Ø IF A$<>CHR$ VAL "13" THEN PRINT AT Y+M,X;" ": NEXT : RETURN

 6ØØ LET POS=IT-L+M: LET D$=C$(POS, TO VAL "1Ø"): GO TO VAL "6Ø1
"+CODE C$(POS,11)
 6Ø1 LOAD /D$
 6Ø2 LOAD /D$ DATA N(): STOP
 6Ø3 LOAD /D$ DATA N$(): STOP
 6Ø4 LOAD /D$CODE : STOP
 6Ø5 LOAD /D$ABS
 6Ø6 LOAD /D$VAL
```

USE ANY KEY TO ADVANCE CURSOR, THEN <ENTER> TO LOAD PROGRAM
USE <CLEAR> COMMAND BEFORE SAVING WITH <SAVE /Ø>
BE SURE TO INCLUDE A SPACE BETWEEN '!' AND '&' IN LINE 1Ø
SEE INSIDE BACK COVER FOR LISTING OF MACHINE CODE USED IN ABOVE

THE OLIGER 2068 FLOPPY DISK INTERFACE

CONTENTS:

OLIGER DISK BOARD "A" ASSEMBLY INSTRUCTIONS

NOTE: ALL parts on this board are installed on the side of the board labeled "component side" and are soldered on the opposite side of the board. This is a plated through board so do NOT install any feedthrough wires. Use care when soldering to keep solder away from the gold plated edge traces on this board.

STEP 1) Install all IC sockets on the board. Be sure the pin one end (typically identified as the end with a small "1" molded into the plastic or possibly a U shaped notch in the center of the IC socket on one end) is installed on the same end as the "1" label etched on the board by each socket mounting area. There are five 14 pin sockets, one 16 pin socket, and one 28 pin socket to be mounted on this board. No socket is to be installed in the U7 oscillator area. Solder each socket pin at every socket on the non-component side of the board and trim excess socket lead length with small "nippy" cutters. Clean all traces of flux from this soldering with acetone or a commercial flux remover solvent and carefully inspect your work. Look closely for solder bridges to nearby traces and unsoldered pins at each socket pad. Touch up anything found not to be a bright and shiny solder connection or any solder bridges.

STEP 2) Install the four lead 8MHZ single chip oscillator in the U7 area with its pin 1 corner (identified by it being pointed rather than rounded as the other three corners are) right above the "1" label etched onto the board. Trim excess lead length from the solder side of the board after soldering.

STEP 3) Install the four 150 Ohm resistors (brown-green-brown-gold) at their locations on the board labeled "R1", "R2", "R3", and "R4". These resistors should have their lead wires bent into a 90 a little from the body of the resistor so that each lead fits into its hole without forcing the resistor down to the board. Be sure that the lead wires do not short to any traces going under the body of the resistor on the component side of the board. Install the 1K (brown-black-red-gold) resistor at its "R5" mounting holes in the same manner. Solder the leads on all the resistors on the opposite side of the board and trim excess lead length from the solder connection.

STEP 4) Install the three 10MFD/6V radial lead tantalum capacitors on the board at their "C1", "C2", and "C3" mounting areas. Be SURE that each capacitor is mounted such that its Positive lead (marked with a "+" on the part and the longer lead of the two) is entering the "+" labeled hole on the board. C3 will need its leads formed outward slightly to fit in its mounting holes. Solder the leads in place on the non-component side of the board and trim excess lead length from the board.

STEP 5) Thoroughly clean all traces of flux from the additional

soldering and very carefully inspect your work completely one more time. Any faults found now can save a LOT of aggravation later in trying to troubleshoot a non-working and possibly damaged board. Only when 100% satisfied that all soldering is good and without any solder blob shorts should you go on to step 6.

STEP 6) Install all small ICs in their respective sockets using the parts list and schematic for correct placement. Be sure that the pin 1 end of each IC (usually identified by a "U" shaped notch or small hole) in installed on the same end as the "1" label at each IC socket. This will result in all ICs except U8 and later U3 having their part numbers towards the right side of the board. U8 and U3 have their part numbers facing the bottom (expansion board) of the board. Verify that each pin on each IC has actually entered its socket contact and not either curled up under the IC or stuck out over the edge of the socket. Install the WD1770 IC (U3) in its 28 pin socket the same way, but use care in handling this one as it is an expensive MOS part and easily damaged by static (take off your shoes!). Your Oliger 2068 Disk I/F "A" board is now complete.

TESTING DISK BOARD "A"

Disk Board "A" cannot be really tested until after Disk Board "B" is also assembled, but you can do a preliminary check of the board by plugging it into your Oliger 2068 Expansion board and connecting its 34 pin "Shugart" type edge traces to a floppy disk drive. This connection is made using 34 pin .1"oc IDC edge connectors pressed onto a length of 34 conductor .05"oc ribbon cable. Keep total cable length below 4 foot. The IDC edge connector cable should be plugged onto the top of the "A" board and the back of your drive(s) so that the odd numbered traces connect to the odd numbered traces on your floppy disk. Now turn on you computer and apply power to your drive(s). The computer should come up normally and your drive(s) do nothing. If your computer crashes on power up or your drive(s) do anything at all, then there has been an error made either in the board's assembly or its connection to your computer.

If the computer comes up normally, key in <OUT 183,1><ENTER>. This should cause the red LED indicator on any drive you have selected to be drive 0 (via its internal dip or jumper strip-see your drive's manual) to light. If this does NOT happen, then check U5 pin 2 for a logic 0. If a logic 0 is found at this pin, then the board IS telling drive 0 to be selected and your drive is either not configured to be drive 0 or is not connected correctly via the ribbon cable. You can try turning the edge connector at the "A" board (after turning off the computer) 180 degrees to see if one of the connectors is plugged in backwards. If one end IS in backwards, no damage will have been done...it

just won't work till it is connected correctly.

Assuming the LED does light, key in: <OUT 143,0><ENTER>. This should cause the selected drive (drive 0) to start its motor and (if a disk is inserted) move its head to track 0. If it does ANYTHING then the board is probably OK and you can proceed with assembling board "B". If no response is gotten then an error is indicated in board "A"'s assembly, the cable connection to your drive, or in the internal set-up of the drive itself. Recheck EVERYTHING and then consult your drive manual to verify that the drive IS set (via its internal dip-switch or shorting header plug) to be drive 0.

OLIGER DISK BOARD "B" ASSEMBLY INSTRUCTIONS

NOTE: ALL parts mount on the side of the board labeled "component side" and are soldered on the opposite side of the board. This is a plated through board so do NOT install any feedthrough wires.Use care when soldering around the gold plated edge traces to keep solder and solder residue from these critical contacts.

STEP 1) Install all IC sockets on the board. Be sure that the pin one end (typically identified as the end with a small "1" molded into the plastic or possibly a U shaped notch in the center of the IC socket on one end) is installed on the same end as the "1" label etched on the board by each socket mounting area. There are four 14 pin sockets, two 28 pin sockets, and one 20 pin socket to be mounted on this board. Solder each socket pin on every socket on the non-component side of the board and trim excess lead length with small "nippy" cutters. Clean all traces of flux from this soldering with acetone or a commercial flux stripper and carefully inspect your work. Look closely for solder bridges to adjacent traces, especially on the 28 pin sockets that use narrow pads and have traces running between the pads. Also check for unsoldered or incompletely soldered pads everywhere on the board. Touch up anything found looking at all suspicious.

STEP 2) Install resistors R1 & R3-R10 (R2 is not used) over their respective "Rn" labels etched onto the component side of the board. These resistors should have their lead wires bent into a 90 about 1/8" from the body of the part, and enter the board without excessive force required to press the resistor completely against the board's surface. Solder in place and trim excess lead length after soldering.

STEP 3) Install two 10MFD/6V radial lead tantalum capacitors on the board at their "C1" and "C2" locations. Be sure that the positive lead (the longer lead of the two) enters the pad labeled with the "+" symbol on the board. Install capacitors C4 (120 or 130PF) and C5 (470 or 560PF) (C3 is not used on this board). Because of lack of available board space, the "C4" & "C5" labels

are beside their mounting holes with a "." between the mounting holes indicating the location. These capacitors are not polarized so they may be inserted in any mannar. Solder these four capacitors in place and trim excess lead length after soldering.

STEP 4) Install the small red Bank Active LED in the two pads under the "ACTIVE" label at the top of the board. Be sure that the LED's Cathode lead (the shorter lead of the two) enters the pad that has a small "C" below it. Install the 1N4148 or 1N914 diode (D1) in its location just above R10 on the upper right hand side of the board, over the standard diode symbol etched marking its place. Be sure that the cathode lead of this diode (banded end) enters the hole that the diode symbol arrow points to. Solder the two diodes in place on the non-component side of the board and trim excess lead length. Remove flux and soldering residues from the board with acetone, nail polish remover, or any commercial flux stripper.

STEP 5) Install pushbutton NMI SAVE switch on the upper left hand corner of the board. Two types of pushbutton switches may be supplied with your kit. The larger switch type (identified by having only two leads) mounts on the component side of the board between the two pads labeled "P1" & "P2" on the solder side of the board. The smaller four legged switch mounts in the four holes between the P1/P2 pads. (If you purchased the Disk 'B' Board as a bare pcb and have trouble locating a suitable switch to mount here, you can obtain this part from John Oliger Co. for $1.00 & a 35 cent SASE.) Solder the switch in place. Install subminiature pc mount slide switch SW1 just right of the NMI SAVE switch previously installed in the three pads between the labels "OK" and "SA" (for OKDOS & SAFE) (You can get this part also for $1.00 & a 35 cent SASE if needed). Solder SW1 in place and slide switch to its right "SA" position for use with JLO SAFE (OKDOS, by Ray Kingsley of Sinware Software, will not be available). If you desire the optional enable/disable switch (SW2, also available at $1.00 & a 35 cent SASE), cut the trace on the solder side of the board between the two upper switch pads in its mounting area. The switch is located just left and slightly above R7 on the upper right corner of the board. Install SW2 and solder in place if it is desired. Now thoroughly clean all traces of flux from the additional soldering and and carefully inspect your work completely once more. Be certain there are no solder blob SHORTS around the parts just added. It's a LOT easier to fix a fault in soldering now then later after the board is tested and found not to work.

STEP 6) Install the ICs themselves. ALL ICs on this board are either CMOS or NMOS devices, and ALL are easily damaged by static electricity. Take off your shoes before proceeding. Install all of the ICs in their correct sockets using the parts list and schematic. Insure that each pin on every IC actually enters its respective socket contact, and does not curl up under the part or

extend over the edge of the IC socket. Each IC should be positioned such that its pin 1 end (identified by a "U" shaped notch or small hole molded into its plastic on one end) is on the same end of the socket as the "1" label etched on the board at each IC socket mounting area. Your Oliger Disk Board "B" is now ready.

TESTING DISK BOARD "B"

Install Oliger Disk Board "B" in your 2068 Expansion Board, SLOT 0, with the computer shut off. Because of the critical timing used on Disk Board "B", I recommend this board's use ONLY in Expansion Board SLOT 0. It does not matter at this point whether Disk Board "A" is installed in the expansion board or not. Switch on the computer while observing Disk Board "B"'s ACTIVE LED. The ACTIVE indicator should flash initially on power up then go out and the computer come up in Basic as it normally does. If the ACTIVE LED should not flash on power up or if it lights and stays lit without Basic regaining control then turn off the computer, remove Disk Board "B", and start looking for errors.

If the ACTIVE LED flashes as it should on power up, then we can now make further tests. This board is enabled whenever ANY Basic error that uses RST08H occurs, so we will check out its operation with a real error. Key in <STOP><ENTER>. The computer should return with its normal STOP report and the Disk "B" board should very quickly flash its "ACTIVE" LED at you. If the computer crashes when this test is tried or if the ACTIVE LED doesn't flash in the least, then turn off the computer, remove Board "B", and start hunting for errors.

Now let's enter a real JLO Extended Basic command. Key in <LET /P=0><ENTER>. The computer should respond with its "OK" report. If it does not, then the "B" board has failed. Best start looking for an error on the "B" board. If the correct report is returned and you have the Oliger or Aerco Printer Port installed, try an LPRINT command or two. Your big printer should respond as the printer currently selected. (NOTE: This new printer support firmware may not work with SOME combinations of Aerco printer I/F and certain printers, but will work with ANY printer used on an Oliger printer I/F.) Next test is to key in <LET /H=1+1><ENTER>. The computer should again return with its "OK" report. If the Oliger Disk "B" board passes all these tests, then chances are all is well and we can now actually try out the complete Oliger disk system. If the board fails, then carefully inspect Board "B" for solder blob shorts, poorly soldered connections, incorrect placement of ICs, etc, and get the problem corrected before proceeding with the test of the actual disk system using both boards and your drive(s). If you simply cannot locate the problem and find you are ready to give up on the board, you do have one

more way to solve the problem. You CAN return the Disk "A" & "B" boards to John Oliger Co. for repair at the flat rate of $20.00 including return postage. This offer is good for all boards supplied as kits or assembled boards whose 30 day warranty has expired. Of course if you purchased these boards preassembled from John Oliger Co., they ARE guaranteed to be free from defects in material or workmanship for 30 days from the date of shipment and if repairs are required they will be performed at no cost to you. However, please write describing the problem before returning for repair. The $20.00 flat fee for repairing the kit version includes all parts and labor required to repair your system EXCEPT the WD1770 chip itself. Because of the high price of this controller chip, EVERY 1770 is tested before shipment and WILL work. So, if this chip is now bad the repair will only include notification of this fact and a complete checkout of the rest of the circuit before its return. Replacement will then be up to you to perform on return of your boards.

OK. We now have tested both boards "A" & "B" as much as we can separately. Now it is time to connect our drive(s) to the system and check everything together.

CONNECTING THE OLIGER 2068 DISK INTERFACE TO YOUR TS2068

Physical connection of the Oliger 2068 Disk Interface to your TS2068 computer is fairly straightforward. For a complete system you will need as a minimum: Oliger Disk Board "A", Oliger Disk Board "B", Oliger 2068 Expansion Board, at least one 3", 3 1/2", or 5 1/4" disk drive with power supply and case, and a cable to connect Oliger Disk Board "A" to your disk drive(s). The drive(s) should be preset to respond to the lowest drive select number possible for the amount of drives you have connected to your system. Thus, if only one drive is used, it should be configured to respond as drive zero. If two drives are to make up the system then they should be configured to respond as drives zero and one, etc. (You may, however, find it desireable in a two or three drive system to have one or more of the drives programmed to respond to two drive select lines for ease in using the MOVE / command. If one drive will respond as drive 0 or 2 and the other as drive 1 or 3, MOVE / will always move from the current drive to the other drive).

Configuring a drive to respond to a certain drive select address number is typically accomplished on the drive's printed circuit board, via a dip switch, shorting blocks, or something similar. Because every brand and model of drive is different, you MUST refer to documentation included with your drive to find what to do to program this setting on your particular drive. While you are checking this, look for notes on other programmable settings for your drive(s). If possible, set your drive so that it will keep the head loaded constantly, light its status LED whenever

the drive is selected, and start its spindle motor whenever the "motor on" input is active. You will find that on "IBM PC compatible" drives these settings will typically be preset correctly, and that you need only program the drive for its drive number. Avoid "IBM AT compatible" drives. These are non-standard drives that actually run more like an 8 inch drive than 5 1/4 inch drive.

Your drive's manual will also state that each drive is shipped with a termination resistor package preinstalled, and that if more than one drive is to be used in a daisy chain that this resistor pack (usually looks like a blue IC) should be removed from all drives except the last in the daisy chain. This "last" drive has nothing at all to do with the drive's selected number, but to the drive that is physically at the END of the ribbon cable connector assembly. Use your manual to find the location of the terminating resistor pack and remove it from all of the drives except the one connected to the end of the cable.

Most all 3", 3 1/2", or 5 1/4" drives on the market today require a separate power supply that can deliver +5VDC & +12VDC at the currents listed in the drive's specification manual. For most 5 1/4" drives this is around .6 Amp @ 5VDC and .9 Amp @ 12VDC per drive. Because only one drive is active at a given time, and the drive's current requirements are much less than this maximum when not selected, you can usually get away with a supply rated at 5VDC/2Amp & 12VDC/2Amp when using two or three drives in a system. Switching type supplies are OK for use to power your drives, but care should be used in physically mounting the switching supply and grounding it. The switcher should not be physically located close to the drive or data cable as the strong RFI it emits can easily cause data errors. A grounded metal shield or distance of at least one foot between supply and drives/data cable will be found to give the desired isolation needed for reliable operation. Another source of problems in using the switching supply can be traced to its required input connection to the green (grounded) conductor of the AC line. Many times, with both the switching supply and the drives themselves (via their screw connection to a metal case which is also connected to this ground) connected to ground together (and thus to each other), RFI can and will radiate through the common ground connection and cause frequent errors and unsatisfactory operation. The cure for this is to disconnect the drives from this ground connection.

Do NOT try to power your disk drive(s) from the computer itself. There is just not enough spare power available to do this. 3" and 3 1/2" drives will typically require much less power than their 5 1/4" counterpart. It is very likely that a little power supply circuit using 7805 and 7812 regulators could be rigged up to power these small drives. (If you are using 3 1/2" drives and recieved the standard 5 1/4" data cable from John

Oliger Co., you can return this for exchange on the header type data cable your 3 1/2" drives require.)

The cable that connects your drive(s) to Oliger Disk Board "A" consists of a 3 foot or so length of 34 conductor .05" on center ribbon cable with 34 pin (considered 17 pin double read out by some manufacturers) IDC edge connectors pressed onto the cable at each end. If more than one drive is to be connected, then the additional edge connectors are pressed onto the cable on the drive end-usually spaced 6" from each other. These connectors are called "Shugart compatible" in that Shugart (this was a very large disk drive manufacturer) started this defacto standard defining the 34 pin connection and what signals are where on this connector. All signals to and from the disk drive are physically on the even numbered edge connector side and the odd numbered side of the connector carries return ground connections for these signals. This results in a ground wire between every signal carrying conductor on the ribbon cable, which tends to improve the performance of the interface by isolating each signal carrying wire from its neighbor. Also, because each active wire in the connector is connected to an open collector output that cannot source any current at all, if the connector happens to be accidently connected backwards absolutely no damage will result to any driver. The Shugart standard actually calls for the IDC edge connector to have a "key" installed between pins 4 & 6 to insure correct hook-up (Oliger Company cables DO have this key installed) but the key is really not required because the interface simply will only work one way and no damage will result if installed backwards. Many interfaces for other brands of computers do not bother to key their board's edge traces or cable. John Oliger Company DOES include these keys on both Board "A" and the optional cables. The keys were originally designed to help avoid errors, so we'll avoid 'em if at all possible.

One more quick note about the drive used with this controller. If the drive used is termed "TRS80 compatible" then chances are you can only have three drives maximum of this type in your system. Radio Shack drives typically use the edge trace IBM compatible drives select their fourth drive with as a "Ready" output from the drive. Most IBM compatible drives do not have a "Ready" output, nor does Oliger Disk Board "A" require one. Thus, you CAN use TRS80 compatible drives but only a maximum of three of these can be used at once and you may have to set the head speed setting (via LET /H=n) different from the default IBM speed and possibly the maximum amount of tracks per side (via LET /T=n) and maximum number of sides (via LET /S=n) per disk. If you have a choice, the IBM compatible drive is recommended.

At this point we should have our drive(s) mounted in a nice commercial cabinet or possibly a handsome homemade cabinet with its own power supply and a 34 conductor ribbon cable connecting our drive(s) to Oliger Disk Board "A". Oliger Disk Board "A" &

"B" are installed in the Oliger 2068 Expansion Board and both boards are completely assembled and ready, with the JLO SAFE (Simple And Fast Extended) Disk Basic in place on board "B". We can now actually try out our new disk system! Before continuing on with how to actually use the system, I suggest that you now refer to the JLO SAFE DISK BASIC COMMAND LIST in the back of this manual and familiarize yourself with new the Extended Basic commands. After reading through the command list, return to the text at this point to actually try out your new system.

NOTE: BECAUSE OF THE VERY HIGH SPEED OF DATA TRANSMISSION THROUGH THE DATA CABLE CONNECTING DISK BOARD 'A' TO YOUR DRIVE(S), THIS CABLE IS SUBJECT TO EMI AND RFI RADIATION AND MANY SAVE/LOAD ERRORS WILL OCCUR IF SUBJECTED TO THESE FIELDS. ROUTE THIS DATA CABLE WELL AWAY FROM MONITORS, TVS, SWITCHING POWER SUPPLIES, OR ANY OTHER DEVICE THAT COULD TYPICALLY EMIT THIS INTERFERENCE. ONE FOOT DISTANCE WILL GENERALLY BE SUFFICIANT FOR THIS APPLICATION. THIS PRECAUTION APPLIES TO THE DISK DRIVES THEMSELVES, ALSO.

TESTING AND USING THE DISK SYSTEM WITH YOUR DRIVE(S)

With Disk Board "B" installed in 2068 Expansion Board SLOT 0 and Disk Board "A" in any Expansion Board slot and connected to your drive(s), power up both the drive power supply and your computer. It should not matter which is powered up first and you may prefer to simply leave your drive(s) powered all the time and only switch off your computer as I do. I DO NOT recommend leaving a disk in the drive during power up and power down because it is possible that a glitch could destroy a file on the disk. You may find that this never happens to you except for that one time that you lost a file that took several days to write, and you did not make a back-up disk.

After the drive supply and computer are both on, the computer should have its copyright on the screen as usual and the drive that is programmed to be drive zero should has its activity LED lit. The activity LED on your drive(s) is used by JLO SAFE Disk Basic to indicate the currently selected drive. The default drive selected on power up is drive zero, hence the reason its indicator is lit now. Now key in <LET /D=1><ENTER>. The activity LED on drive zero should extinguish and if you have a two or more drive system, the drive programmed to be drive one should now have its activity LED lit. If any of this does not happen, then stop now and find out why. Most likely thing to cause problems in this area is the programming of the drive itself. Recheck your drive manual for the required settings on the drive as detailed earlier and check the drive to be sure it is set correctly. If the problem does not appear to be in the drive(s), then recheck Disk Board "A". This is the board that actually enables the drive hardware wise. Disk Board "B" is the firmware controller board

that handles the Extended Disk Basic commands and Disk Board "A" is the hardware interface actually used to control and communicate with the drive(s).

At this point you should use the <LET /D=0><ENTER> command to reselect drive zero. If your drive(s) is anything but the default IBM compatible type, reconfigure the default settings as required by your drive(s). The default IBM settings are: Tracks=40, Sides=2, and Head step speed=0. Use the LET /T=n, LET /S=n, and LET /H=n commands to configure the system as required by your particular drive as needed. Note that Amdek 3 inch drives are actually 40 track single sided drives, despite what their owner's manual states. To get the other side on the Amdek drive you must physically turn over the disk. Each side of these 3 inch disks can be treated as a separate disk, each requiring individual FORMATTING. Now we will actually use a drive.

Insert a blank disk into drive zero and key in <FORMAT /"TEST"><ENTER>. FORMAT is obtained on the 2068 by getting into extended mode by pressing cap & symbol shift together and then pressing shift zero. After <ENTER> is pressed to start the FORMATTING process, the current drive should start. After a one second delay to allow the drive's motor to get fully up to speed the drive should actually start formatting the disk.

In formatting the disk, you should hear the head being stepped at a rate that will vary depending on whether you are using single or double sided drives. SAFE will format all possible sides of the disk and then verify that the format was good on both sides before proceeding to step in and format the next track. The head is stepped no more than is actually required to do the job, keeping wear and tear down on the moving parts of the drive. After the entire disk is formatted, SAFE will restore to the very first track to initialize the disk's CATalog.

If the entire disk formats correctly then when done the computer will return with its "OK" report, along with details on the screen showing disk capacity, etc. (an auto-CATalog is performed). If ANY errors are encountered during FORMATing, a "TOOT" will be sounded and a return will be made with the "DISK I/O ERROR" message. If an error occurs during FORMATing, the most likely problem is the disk itself. Perhaps the disk has a scratch or foreign matter on its surface and because of this is defective. You can try to format the disk again, but if it keeps giving errors then you are advised to toss the disk and use another. A "trick" that sometimes works on marginal disks when trouble is encountered in FORMATing is to run a small permanent magnet all over both sides of the disk. Be careful not to allow the magnet to actually touch the disk surface in the cut-out areas. Be very carefull that you do not get the magnet around any recorded disks or cassettes you may have laying around or some valuable files can be easily wiped clean! This magnet erases the

test tracks normally put on the disk at the factory by the manufacturer. These tracks have been found to cause initial problems in FORMATing on occasion.

Another common cause of errors in initial format can be traced to noise being picked up by the drive data cable, the drive itself, or via the grounds used on the drive/drive housing. The data cable and drive should NOT be physically located close to any source of RFI or EMI. The most common source of this interference around a computer is the monitor or television itself. Either separate the drive and cable from the monitor/tv or shield them from the noise with grounded metal. Look also for ground loop noise problems causing disk errors in formatting, especially on a new installation. Remove or install grounds as required on initial installation to get reliable error free operation of the system.

Because JLO SAFE Disk Basic always operates in double density mode, you MUST use certified double density disks with this system. If your drive is a double sided drive, then you should be using double sided double density disks. If your drive is a "quad" density 80 track per side unit, then you should be using 96TPI double sided disks. The use of "bargain" disks may work just fine for unimportant applications, but if your application is important and you do not want errors to occur at any time if at all possible, then use only high quality double density disks, either single sided, double sided, or 96TPI double sided as required by your drive.

There is one more point I would like to make concerning FORMATting disks using the Oliger 2068 Floppy Disk I/F w/JLO SAFE Disk Basic. Unlike most disk I/Fs, you can FORMAT a disk at any time desired without any Home ram needed by the command. Many disk formatters require several thousand bytes of your memory for use as a large buffer. JLO SAFE Disk Basic FORMATs in a way completely unlike most FORMATters, and uses NO ram FORMATting this way. Thus, if you have a very large program in memory and find you do not have a FORMATted disk handy to store it on, you can ALWAYS FORMAT immediately and will NEVER get an "OUT OF MEMORY" error when you do!

At this point we have successfully FORMATted a disk and we are ready now to actually SAVE and LOAD to disk. As a test, key a few Basic lines into the computer, such as:

```
10 REM This is a test of the Oliger Disk I/F
20 CLS: PRINT AT 10,10;"WORKING"
30 BEEP 1,1
```

Now from Immediate Mode key in: <SAVE /"test prgm"><ENTER>. Your drive should start and, after a one second pause for the drive motor to get fully up to speed, should SAVE

the the little test program to disk as a BASIC file named "test prgm". Remove the disk from your drive, reset or switch off then on the computer, and reinsert the disk. Now key in: <LOAD /"test prgm"><ENTER>. The drive should start up and quickly LOAD back in the test listing originally SAVEd. Now let's reSAVE this little test program to disk but make it auto-run on LOADing. Key in: <SAVE /"test prgm" LINE 1><ENTER>. After the drive startsand SAFE reads the CATalog of the disk, the computer will "TOOT" and print a "FILE EXISTS!" message in the lower part of the screen. What SAFE is saying is that a BASIC file with a name of "test prgm" already exists on this disk (we just SAVEd it!), and if you do not desire the old file to be overwritten by the new file, to "BREAK now or forever lose the old peace (file)". If you have not pressed shift/break in about 5 seconds, or if you acknowledge that you DO want the old file overwritten by pressing the ENTER key, SAFE will over-write the old file with the new file if it will physically fit in the space allotted to the old file. If you press shift/break SAFE will return with the BREAK error mssage, or if the new file is physically to large to fit in the space allotted to the old file SAFE will return with a FILE TOO LARGE error. In either case the old file on disk will still be intact.

Let SAFE overwrite the old test file with the new BASIC file and then test that the LINE extension given with the new SAVE will really work by LOADing in the test program with the command <LOAD /"test prgm"><ENTER>. After the file is LOADed, it should auto-RUN and show it has by printing "WORKING" and then BEEPing. As you can see, these commands work exactly like their equivalent cassette based SAVE/LOAD commands, and in fact you can usually edit a program using the cassette SAVE/LOAD commands by simply adding a "/" after the SAVE or LOAD portion to have the computer SAVE/LOAD to and from disk rather than cassette. One exception to this is a <LOAD ""> command. You MUST give a filename when LOADing from disk because there is not a "next program" to LOAD, as there is on a cassette tape.

The CODE, SCREEN$, DATA, and DATA$ SAVE/LOADS also work exactly like their equivalent cassette based commands do, but you now have two additional types of SAVE/LOADs with JLO SAFE V2. The first "new" SAVE/LOAD is SAVE /"filename" VAL or LOAD /"filename" VAL. This new SAVE/LOAD will SAVE or LOAD all the variables in the current Basic program, but not the program itself. When LOADing a VAL or VRBLS file, the current Basic program's variables are NOT cleared purposely. This allows more flexibility in using the command. If you do not need the current program's variables, then simply CLEAR before you LOAD the new variables.

The other "new" SAVE/LOAD added to TS Basic by JLO SAFE V2 is SAVE /"filename" ABS or LOAD /"filename" ABS. This new SAVE/LOAD will SAVE or LOAD the total STATE (everything!) of the computer to or from the disk. This includes the current screen, Basic program, variables, machine code, aternate character sets, etc.

all in one big shot! This type of SAVE is used by JLO SAFE V2 whenever the NMI SAVE feature is used. A good typical example of using the ABS "STATE" type SAVE/LOAD is when using HOT Z 2068 or HOT Z AROS. If you quit to Basic and enter <SAVE /"user file" ABS : RANDOMIZE USR 24098><ENTER> you will SAVE the machine code you have been working on, the current Basic program and its variables, the HOT Z name file that you use with your machine code program, and HOT Z itself if you are using the home ram based version. ALL THIS IN ONE SHOT! Thus you can SAVE EVERYTHING to disk quicker than you could figure out where the name file was at and its length before! By the way, JLO SAFE was written using Sinware's HOT Z AROS V2.08. I very highly recommend HOT Z AROS to machine code buffs.

Now, lets go on and try out the <SAVE /0> and <LOAD /0> or simply <LOAD> commands. This special SAVE/LOAD is intended to store a user written menu as a Basic program on track 0, sectors 1 to 3. To try out this command, enter the following Basic program:

```
10 INK 7: PAPER 0: BORDER 0: CLS
20 PRINT AT 1,1;"PRESS KEY FOR DESIRED PROGRAM"
30 PRINT AT 6,0;"1: For Space Zap"
40 PRINT '"2: For Nomad"''"3: For Hungry Hippo"''"4: For Chess"
50 IF INKEY$="1" THEN LOAD /"Space Zap"
60 IF INKEY$="2" THEN LOAD /"Nomad"
70 IF INKEY$="3" THEN LOAD /"Hung Hip" ABS
80 IF INKEY$="4" THEN LOAD /"Chess" ABS
90 GOTO 50
```

This program demonstrates a simple file 0 menu for a disk that has only 4 files in use. SAVE this program now with <SAVE /0><ENTER>. After the SAVE is complete, reset the computer and key in: <LOAD><ENTER>. The file 0 menu should quickly LOAD in and run, awaiting the press of keys 1 to 4 to LOAD in the file desired.

It should be noted here that this special file 0 cannot exceed approximately 1 1/2K in length and that it will ALWAYS auto run when LOADed. If you do not desire the program to auto run then make the first line of the Basic program a STOP command. Use of a LOADing menu with file 0 to select the regular file desired is a big part of the real power behind this disk system. You can have your program selected and running before you can even get the long-winded commands required by other 2068 disk systems keyed in!

THE CATalog COMMAND

Whenever a file is SAVEd using JLO SAFE V2, a copy of the files name, its type, its length, and other information

concerning the particular file is stored in a special area on the disk called the disk's CATalog/directory area. This information is physically located on the disk at track 0, sectors 4 through 10 (3 1/2K). Updates made to the disk's CATalog are done by SAFE automatically whenever a file is SAVEd or a filename changed with the RESTORE / command.

The JLO SAFE V2 user can at any time access the information stored on the disk's CATalog by using the CAT command. CAT differs from a file 0 menu program in that CAT will only display information as to the current disk's contents of the screen, while file 0 is an actual user written program that can very easily be LOADed in by the user and can actually perform the task of finding the user's desired file and LOADing it for him, etc. (file 0 is physically located on side 0 track 0 sectors 1-3)(1 1/2K). A file 0 menu may use the CAT command itself to display the information the user requires to select the desired file, or it could contain the information on the disk's programs within itself with perhaps more details given than is possible using the CATalog command. The CATalog command may be used when writing a file 0 menu program.

The CAT command will detail information on the screen that shows just exactly at what track/side settings the current disk was formatted, the disk's name, its capacity, how much of its original capacity is still free for use, the total number of files, every filename, every file's type, every file's length both in 5K "CYLinders" (used in allocating disk space to hold the file) and in actual bytes, and the starting address for BYTES files or starting line number for auto-running BASIC files saved with the LINE extension. It must be remembered that the CATalog is just INFORMATION concerning the contents of the disk while a file 0 menu is a program unto itself that can "boot" the desires of the user as he wishes. JLO SAFE was written to NEVER "AUTOBOOT" on power up for good reason. Older DOSes HAD to "BOOT" from disk in order to run at all, because they were ram-based mc programs; There was no resident DOS aboard the computer on power-up. Thus every time the computer is powered-up on a system like this, the computer will ALWAYS go through its "thing" hunting for this or that on the disk, making the user insure that the disk IT wants is in the correct drive, and just generally slowing down the whole process of simply turning on the computer. SAFE does have control of the 2068 on power up, but it only quickly initializes itself and lets go of the control, leaving all the options up to the user as to what he wants done when he turns on his machine. He can still very simply "BOOT" a file by using a very easy file 0 <LOAD> command or he can do anything else that he may desire at that point. In other words, SAFE was written to keep the computer the slave of the user, rather than the user the slave of the computer.

THE ERASE COMMAND

Any file that has been saved on the disk using one of the SAVE commands can be removed from the disk using the ERASE /"filename" command. It should be noted, however, that ERASE using JLO SAFE is different from ERASE with any other DOS in several respects.

When SAFE erases a file on the disk, it physically reclaims the disk space used by the file. It moves all files above the file to be erased down on the disk to close the gap. JLO SAFE´s ERASE must work this way for several technical reasons. The end result of this for the SAFE user is some advantages and some disadvantages in comparison to other DOSes. I will detail these;

Because disk space is physically reclaimed by SAFE when ERASE is used, the command itself takes some time to actually execute. If the file to be erased is close to the beginning of the disk (towards the top when listed with CAT), SAFE will take a lot of time to erase it. It will have to move every file on the disk after the file to be erased down on the disk. If the file is towards the end of the disk (near the end when listed with CAT), SAFE will be very fast in erasing the file. If the file is the last file on the disk, the erase will be immediate. Thus it can be seen that the amount of time used by erase will vary depending on the location of the file to be erased.

Because erase can take some time to execute, the chance of the disk being corrupted is much higher than when using other SAFE commands. The longer a disk is turning under control of the disk controller, the more likely it is that a glitch, brown out, power failure, etc., is to happen. Some precautions are taken by erase to keep loss of data to the least possible, but if power would suddenly die right in the middle of an erase, you COULD lose up to ten files on the disk. If an error or crash/power problem is encountered when using erase, load all files to can still get back into the computer, resave them on another good disk, and reformat the corrupted disk.

Thus the best advise is to BACK UP all important files on another disk. If its important and you don´t have a back up disk then don´t use ERASE on the disk. It´s simply too risky. ERASE is like most things with floppy disks; It will work fine for years without any problems at all, until one day something happens and several very important files are lost. You can get too confident and the odds are this will eventually lead to problems (on ANY disk system!).

The advantages (remember, I said there was some of these, too!) of SAFE´s ERASE working this way are; Because space is physically reclaimed on the disk, once a file is erased it will REALLY be as if it had never been put on the disk in the first

place. Other DOSes don't really erase anything, they simply mark the space used by the file as being open and remove the CAT entry. These DOSes use what is called an allocation table that marks every cylinder on the disk as being either available or not available. Because SAFE does not use an allocation table at all, it is much more flexible than most DOSes. This is how SAFE can allow tracks from 10-255 and single or double sided drives, etc., and all via a simple command from Basic w/o using "config files" or having to specially order the DOS eprom to support a certain type or size drive.

Also, because this space is reclaimed by SAFE, your disks are not left with "holes" in its filespace. It is impossible to "fragment" a SAFE disk as will happen with just about every other DOS for any computer. Depending on how another DOS actually works, this "fragmentation" is caused by the holes left in erasing files and will either cause problems such as "DISK FULL" reports when a file SHOULD fit in the space left on the disk or in extended load/save times as the DOS has to store or retrieve files in pieces scattered all over the disk. Remember that if another DOS is spending more time accessing a disk for save or load, because it is scattered around in pieces on a fragmented disk, then it is more likely to corrupt the disk doing this than SAFE is. Again, the longer a controller is actually controlling the more likely it is for something to be corrupted.

To sum all this up about ERASE:

1) Always back up important files before using ERASE.

2) SAFE allows updating a file by re-saving the file under the same name if it will physically fit into the space allotted for the file. You can reserve extra space by first saving a file substantially larger on the disk and then resaving the real file under the same name. It is far preferable to use these options instead of ERASE for files you know will grow and require updating. You will only then need ERASE to clean old obsolete files from the disk.

3) When erasing more than one file, always erase backwards. (IE: ERASE the files listed last in a CAT listing first)

4) The time taken for an ERASE will vary depending on its placement in a CAT listing. The lower a file is listed the faster the file will erase.

5) Once a file is erased, IT IS GONE FOREVER! Be SURE you want the file erased before you tell SAFE to do it! Once the critical portion of ERASE is under way, the BREAK key is disabled.

CORRUPTED DISKS???

Occasionally you might come across a disk that has become corrupted in its file storage/pointer area (track zero) that will simply refuse to load any file at all. What will usually cause this is a glitch on power-up of either the drive or computer, with the disk installed. Occasionally a bad WD1770 controller chip has also been found to cause this problem. ALWAYS remove your disk from your drive when powering up either your computer or drive unless the disk in the drive is something that you do not care if it is destroyed (such as a game disk that you have made a good back-up from). It is best to simply get into the habit of doing this all the time, regardless which disk you have installed.

But EVERYONE can forget sometimes so I will present a small program that MIGHT bring your disk back to life, or may kill it completely. If it's gone already, you have nothing to lose to try it and you just might be able to recover the data from the corrupted disk!

The FIRST thing to try if you have a corrupted disk is to re-save file zero. File zero uses sectors 1 through 3 of track zero, and when SAFE loads ANY file it ALWAYS loads all ten sectors of track zero. Because the glitch could have only corrupted one of the first three sectors, we have a 30% chance that simply re-saving file zero will fix the problem. Enter a single line program such as: 10 PRINT and save it to file zero with the command <SAVE /0>. Now try a CAT or try to load a regular file. Perhaps we have fixed the disk already! If not...

If not, then after the computer stops with its error T report, enter the command <PRINT IN 175>. The sector number that is causing the problem will be printed on the screen. Because the CATalog area starts with sector 4 and ends with sector 10, and re-saving file zero into sectors 1 through 3 did not fix the problem, you will find that the actual bad sector will usually be in the range 4-10. If the sector giving the problem is NOT sector 4, then it is VERY likely that we can restore the catalog area by using a little utility program. If it IS sector 4, you can try it too...it STILL could work!

The utility program is given below. Key in the program as listed, do a CAT (even though it stops with an error), then run the program.

```
10 FOR N=32000 TO 32006: READ A: POKE N,A: NEXT N
20 RAND USR 32000
30 DATA 205,10,0,205,142,10,223
```

This little utility program re-saves the CATalog back onto the disk. If the error was actually in a later portion of the

catalog area that was not being used (the CAT area can hold 177 files, and will not be full unless you have this many files on the disk!), then the data re-saved will be correct and the disk will be completely recovered. You will find that over 90% of the time this program will restore access to your disk.

To those interested in what this program does at the mc level, I provide the following annoted assembly listing. It is a very simple little routine.

```
CALL 000A     ;turn on Oliger Disk B bank
CALL 0A8E     ;save the contents of the B bank buffer to track 0
              ;Data will be put there by using CAT before
              ;running this program.
RST 18H       ;turn off B bank, turn on int. & ret to Basic
```

PROBLEMS USING THE SAFE V2 "MERGE" COMMAND

If you have a program that will not MERGE with another program, chances are the reason it will not can be traced to the program itself. It has been found that some programs can become corrupted in such a way that MERGE will act like it has worked, but when you look at the Basic listing for the appended Basic code it will not be there. The reason for this has been traced to be usually caused by corruption at the end of the Basic program itself.

Because of the way Basic programs are stored, saved, and loaded on the 2068, once something like this occurs and the program re-saved (using either cassette or disk), it will forever exist because as far as the computer is concerned it IS a part of the Basic program, even though it can't use it or LIST it.

So, how do you know if a program has this problem? If you cannot use the SAFE MERGE command with it then it is very likely. But the REAL test is to use the command <PRINT FREE> immediately after power up and again after the program is loaded and the removed using <CLEAR :DELETE ,>. If the numbers don't match you can be fairly certain that the program does contain corruption (unless it had self-run and used the CLEAR N command). I will now detail step by step how to fix this:

1) FIND AMOUNT OF PROGRAM MEMORY CORRUPTED:

A) Write down result of the statement <PRINT FREE> immediately after computer power-up. (38652 on standard 2068 w/o any cartridges installed)

B) Load corrupted program. Save variables to disk with the command <SAVE /"vars" VAL>. Clear both program and variables from the computer with the command line <CLEAR :DELETE ,>. Now write down result of the statement <PRINT FREE>.

C) Subtract the number written down in 1B above from the number written down in 1A. Write this number down. This is the

amount of memory corrupted.

2) FIND CORRECTED BASIC VARIABLES POINTER:
A) Re-load corrupted program and remove the variables again with the <CLEAR> command.
B) Enter the command <PRINT PEEK 23627+ PEEK 23628*256>. Write down this number.
C) Subtract the number obtained in 1C from the number you just wrote down in 2B. Write down the result.

3) FIND NEW VALUES TO POKE:
A) Divide the number obtained in 2C by 256. Round down to the next whole number and write this down. (MSB)
B) Multiply the number just written down in 3A by 256. Subtract this product from the number obtained in 2C. Write down the result. (LSB)

4) SET THE VARIABLES POINTER CORECTLY AND CLEAR THE CORRUPTION:
A) Set the variables pointer with the command <POKE 23627,LSB: POKE 23628,MSB> with "LSB" and "MSB" replaced by the numbers written down in 2B and 2A respectively.
B) Clear the corrupted memory by using the <CLEAR> command again.

5) GET IT ALL BACK ON DISK:
A) Re-load back in the old variables with the command <LOAD /"vars" VAL>.
B) Re-save the repaired file with the regular <SAVE /"Filename"> command.

The "cleaned-up" program should now allow MERGE to operate and be shorter to boot!

WHAT HAPPENED TO THE "*" SEPARATOR LIKE THE I/F ONE USES?

About this point (or maybe even before now..) you are probably wondering what happened to the commands that use a "*" as the disk identifying token. After all, initially I had said that this interface would use this character in a way similar to how it is used on the Sinclair I/F One. At first JLO SAFE Disk Basic DID use the "*" identifier, but I had to later change the disk token to a different one for a very good reason; because Timex actually interprets the "*" character after a SAVE/LOAD command itself within the Timex Home Rom! This totally defeats the purpose of the disk command token in the first place, which is to cause a RST08h error Call with the Basic System Variables pointed at the disk token itself (because IT caused the error) so that further interpretation of the line can be made from this concrete starting point. Timex was probably trying to plan for the future when it decided to make this change, but it certainly made interpretation of such statements considerably harder, especially when you consider that the Spectrum Rom WILL NOT

interpret the "*" token and will stop where expected. Thus was born the "/" Oliger Disk Basic System separator/identifier. It was only one key over, easy to get at, and had exactly the opposite meaning the "*" character has.

So now we have the "*" character used on the Timex and Sinclair systems, OPEN#4 used by another, CATALOG & MOVE used by a third (aren't you glad WE don't have to CAT to LOAD?), and the "/" character used on the Oliger System. Isn't it nice that everything is so compatible?

USING THE PUSHBUTTON NMI SAVE FUNCTION

The Pushbutton NMI SAVE Function used by the Oliger 2068 Disk I/F w/JLO SAFE is a very desireable feature when upgrading to a disk system from a cassette based system. Chances are the 2068 owner upgrading to this disk I/F already has a lot of money invested in cassette based programs for his computer. Converting all of these programs himself to work with a disk system is a very large if not impossible task. The programs were designed for LOADing by cassette and may in fact have all sorts of copy protection installed to keep the lawful owner from making a copy of the cassette either on another cassette or a diskette! This protection was installed NOT to keep you from using the software on your new disk system or to keep you forever locked into waiting on cassettes to LOAD with all the usual "TAPE LOADING ERROR" reports, but simply to keep a few people from making some illegal bucks by selling something that isn't theirs to sell.

JLO SAFE allows you to store 99% of your cassette library on your disks by simply LOADing in the program one time from cassette and then SAVing it to disk by using the NMI SAVE pushbutton. To use this feature, you simply LOAD the program from cassette, press the NMI SAVE pushbutton, and after an acknowledge "toot" pressing one on the top keyboard keys "1" - "0" to name the ABS STATE type file the character the key represents. You can later change the name to any you desire by using the RESTORE / command from Basic after looking at the CATalog entry name using the CAT command. When you want to reLOAD the program later, you do so using the command LOAD /"filename" ABS.

Another function of the NMI pushbutton and JLO SAFE is to allow screen dumps to be sent to the Oliger 2068 Printer I/F at any time, via the press of the COPY key (key "Z") after the NMI acknowledge TOOT. For more details on this function see NMI SAVE and COPY / in the JLO SAFE V2 COMMAND LIST in the back of this manual.

NOTE: When using the NMI SAVE to disk function, care should be taken NOT to INTERRUPT at certain critical times. NEVER press the NMI SAFE pushbutton when the Board "B" ACTIVE LED is lit or when SAVing or LOADing to cassette tape (because the exrom is

active during cassette saves and loads). I suggest playing around with the button when nothing important is currently loaded to get the feel for the procedure. Remember, too, that Disk Board "B" is supposed to be installed in Expansion Board SLOT 0.

WRITE PROTECTING YOUR DISKS

When the write protect notch of your disk is covered with tape or the little covers supplied with most brands of disks, this interface cannot write anything to the disk, period. I strongly suggest that after a particular disk is full of programs that this notch be covered to protect the contents from "accidents". Keeping these notches covered all the time and removing them only at the time of a SAVE may be a wise decision, too. Beware that if the notch is covered and a NMI SAVE is attempted, a crash will very likely result because most "protected" programs have fiddled with the 2068/Spectrum system variable ERR SP to make any error cause a crash rather than a return to Basic. JLO SAFE will at least attempt to return to Basic with a "DISK I/O ERROR" message.

TECHNICAL DETAILS ABOUT THE OLIGER DISK I/F AND JLO SAFE

(Gory details for the "Hacker" only..)

Oliger Disk Board "A" is the real disk interface board in this system and Oliger Disk Board "B" can be thought of as a Basic language expansion board containing its own 8K of eprom and 8K of ram space. JLO SAFE Disk Basic is a firmware program written by John Oliger to extend 2068 and Spectrum Basic in support of the real disk interface Board "A". Commands added via use of Disk Board "B" do not really need to have anything at all to do with the disk interface proper. An example of this is the LET /P=0 command, which ties firmware within the "B" eprom to the regular printer channel, allowing instant use of the Oliger or Aerco printer I/F without a LOAD required. Disk Board "B" could in fact be used by itself without Disk Board "A" and the printer support would still work correctly. It would be possible for the user himself to add additional commands in the same manner that JLO SAFE does. The JLO SAFE Disk Basic disassembly may be used as an example as to how to do this by the experienced mc programmer or by somebody who simply wants to see what he can do. The following details concerning Disk Board "B" along with the JLO SAFE disassembly should be a valuable aid in this type of endeavor:

ANY opcode fetch (M1 NOT active) from memory locations 0008-00Fh or 0068-006Fh will turn on or enable this "B" bank of eprom/ram. Any NON opcode fetch read from this same area (M1 NOT inactive) will cause this "B" bank to be switched off or become

disabled. (Actually, a memory write to this region would also disable the "B" bank, too, but this is not recommended because doing so would conflict with use of the Oliger 2068 Eprom Programmer) If the instruction JP 1000 were at location 0008 in the "B" eprom, the "B" bank would turn on when the C3 opcode for JP was fetched, then turn back off again when the 00 and 10 data bytes were fetched, because M1 NOT would not be active when these two bytes were fetched. If mc instructions that do not require data bytes are used, the bank stays on and the instructions are executed correctly. This is why there is a RST28H at location 0009 in the SAFE eprom. A CALL or JP instruction could simply not be used in this critical area from 0008-000Fh (or from 0068-006Fh). To turn off the "B" bank, the instruction LD A,(0008) is generally used at the lowest level of returning to the regular bank of memory, usually with a PUSH AF preceeding this and a POP AF, EI, and RET located in the HOME and Spectrum roms following the LD A,(0008) instruction in "B" bank. The very next instruction executed after LD A,(0008) will be in the bank of memory currently active underneath the "B" bank.

When the "B" bank is enabled, it has higher priority than any other bank of memory on the 2068 computer including 2068 memory expansion boards available later. When the "B" bank is disabled, the next highest priority bank will regain control. This will normally be the Dock (if using a Spectrum Emulator or hopefully other LROS cartridges that modify the operation of the 2068) or Exrom (but it is usually disabled via OFEX in SAFE if it was active) and if neither of these are active then the Home bank will normally be enabled.

These last details were given only to explain the lowest levels of operation when in the "B" bank. Using JLO SAFE, the entry points at 0008-000F and 0068-006F are already taken care of, and RST18h and RST20h can readily be used to jump to or call routines in the Spectrum, Timex, or Exrom. To use RST18h, you need only insure that the bank desired to return to is currently active under the "B" bank (via ports F4 & FF), PUSH the desired address to be jumped to on the machine stack, and then perform a RST18h. The restart will get control of the computer to the address pushed in the other bank, reenable interrupts (interrupts are always disabled when the "B" bank is active), and clear the stack back to the word prior to your PUSH of the destination address. All registers will be preserved in doing this. To use the RST20h CALL Timex, Exrom, or Spectrum rom routine, again you need only push the desired address to CALL on the stack, insure that the bank desired is prepared to be active when the "B" bank is turned off, (via ports F4 & FF) and perform the RST20h instruction. Interrupts will be enabled, control passed to the desired address in the next lower priority bank, (normally the Timex or Spectrum Rom) and on the next RET instruction the "B" board will again gain control, interrupts disabled, and control passed to the next address following your RST20h instruction. All

registers will be preserved both to and from the destination routine. If an error occurs in the Basic rom called, control will NOT normally be returned to the "B" shadow bank.

Any routine contained within the SAFE eprom can be accessed from outside the 0-16K area by simply turning on the "B" bank using CALL 000A, the desired routine accessed directly via another CALL or the "B" bank's ram manipulated as desired, and then turning off the "B" bank by using the RST18h instruction instead of the RET instruction when done. This way, interrupts will be disabled and then reenabled when finished as required by the SAFE eprom.

Regarding the main disk interface Board "A", this circuit is a very straightforward implementation of a WD1770 based disk interface board. The WD1770 data sheet is really necessary to even start to understand how to communicate with and deal with this somewhat intelligent disk controller chip. I/O port 8Fh is used to read the status register and write to the command register of this chip. Port 9Fh is used to read and write to the 1770's TRACK register, port AFh reads/writes to its SECTOR register, and port BFh reads/writes to its DATA register. The last port used on this interface is port B7h, which is a "write only" port. Bit 6 of port B7h is the double/single density select bit. A zero here selects double density mode and a one will select single density mode. JLO SAFE ALWAYS uses double density mode. Bi 7 of port B7h selects the side of the disk currently active on a double sided drive. A zero here will select side zero and a one will select side one. Bits 0-3 of port B7h selects drive 0-3 respectively if high. Bits 4 and 5 of port B7 are not used, nor is the INPUT port B7 at all. ALL ports used on Oliger Disk Boards are fully decoded. I believe in following the computer manufacturer's lead in this area. Timex fully decoded all of its I/O ports on the TS2068, so ALL I/O products for the 2068 from John Oliger Co. are and will be fully decoded.

JLO SAFE Disk Basic FORMATS all disks to have ten 512 byte sectors per track. This will make a standard IBM type 40 track double sided drive have 400K formatted capacity. SAFE uses 5K of the storage for file 0 and the disk's CATalog, so a 40 track double sided disk would end up with 395K of free disk storage area available for use. On many drives you can actually format at a slightly higher maximum tracks than the drive is rated at, and if this is done SAFE can and will use the extra space. You can do this by using the LET /T=n command before a FORMAT / command. I can personally get 43 tracks from my Tandom TM100-2 drives leaving 425K of free formatted space on a disk after a format and 84 tracks from my CDC "80 track quad density" drives leaving 835K of free formatted space after a FORMAT. The number of tracks a particular disk is formatted at is stored in the CATalog area, so the system will know the number of tracks available on the disk in future use of this particular disk without using the LET /T=n

command again.

Disk space is allocated in 5K (the size of one track on one side) "CYLinders" by JLO SAFE. JLO SAFE does not use a bit mapped allocation table, but stores pointers into the file area within the CATalog area itself on track 0. Disk storage efficiency is slightly sacrificed for increased speed and simplicity by JLO SAFE. This results in a faster LOAD/SAVE time than most DOS and less chance of complex "bugs" to be found by the user under certain untried conditions because the structure to so much simpler than many Disk Operating Systems.

In a ABS state type SAVE/LOAD, 48 1/2K of data is actually sent or retrieved from the disk (3E00 to FFFFh). Thus the last 1/2K of ram in the "B" bank is saved to disk along with the actual file. Anything that may be needed upon reLOADing, such as Stack Pointer, is copied into the area of "B" ram that will be SAVEd before the actual transfer begins. During a NMI SAVE, ALL registers are stored in this area along with control flags and the contents of video control port FFh. This information is used when re-LOADing to return the state of the machine to be exactly as it was when originally SAVEd.

There is no sector to sector "SKEW" on a disk formatted with JLO SAFE Disk Basic, but there is a track to track sector skew of three to allow for head settling time when the disk steps in to the next track within a file. Because each sector takes 20ms to pass by the head, this will give 40ms + header and intergap time for the drive to step in and settle. JLO SAFE allows with software a minimum of 30ms for the head to settle, along with the actual head step rate time unused in stepping the head itself. If the slowest head step rates are used, the disk will have already passed the beginning sector and another entire revolution of the disk will be made before starting to LOAD from a new track. This means that with slow head step rates set via the LET /H=n command, 200ms+ is actually allowed for head settling time. If a drive isn't fast enough to handle THIS you should certainly start looking for another drive! JLO SAFE will ALWAYS restore to track zero and then step back out to the required track when LOADing or SAVEing. This makes any mechanical movement of the head when the controller is not directly in control redundant, and adds very little to the time required to access the disk itself. The biggest "time waster" in LOADing or SAVEing to disk is used in allowing for disk spin-up (so that the disk is sure to be running at the correct speed without "wow" from its speed controller circuitry). JLO SAFE uses the WD1770's own option to allow 6 revolutions of the disk (a little over a second) before doing anything unless the disk is already spinning. This very generous amount of time should be sufficient for even the most archaic of drives to be found today.

Files are arranged on the disk in logical order. The

CYLinders are accessed on alternating sides of the disk on a double sided drive to keep head movement to the minimum required to store or retrieve the data. Files are physically located on the outermost cylinders available and space is allocated working into the center of the disk. The effect of this is that the most reliable space (the outer part of the disk) is used first before any use of the less reliable inner tracks is made. Unless a disk is completely full, the very innermost tracks (which are the most unreliable) are never used.

HOW TO ACCESS SELECTED SAFE V2 ROUTINES FROM MACHINE CODE

(More gore for the Hacker!)

Several of the more important commands within JLO SAFE V2 can be accessed at the machine code level if done in the correct manner. These include the CATalog command, the ERASE command, and all of SAFE V2's SAVE/LOAD/MERGE commands. This is how to do it.

HOW TO CATalog FROM MACHINE CODE

1) Enable the B Bank of eprom/ram by using the instruction CALL 000A. This will turn off interrupts, turn on the B Bank and then return control to your routine.

2) Perform the CATalog function by using the command CALL CATL @1906. Alternately, if you KNOW the catalog data has already been loaded into the BRAM buffer by previous use of other disk related commands, you can call CAT2 @1909 which ill skip the actual loading of the information from disk into the BRAM buffer. The CATalog command always uses channel #2 so if you desire use of a different channel after using the function then your program should re-open the desired channel.

3) Turn off the B Bank of eprom/ram by replacing the RET instruction in your subroutine with the RST 18H Z80 instruction. RST 18H will turn back on the maskable interrupts, turn off the B Bank, and then return to the last location pointed to by the (SP) just prior to the execution of RST 18H. If it is desired to perform other functions using the B Bank after a CATalog, you can leave it enabled and do so, remembering that interrupts ARE disabled and must stay that way while the B Bank is active. If you do not desire a RET made, you can use the two instructions LD A,(0008) EI to replace the RST 18H and they will turn off the B Bank and then re-enable maskable interrupts.

HOW TO SAVE, LOAD, ERASE, OR MERGE TO DISK FROM MACHINE CODE

1) Turn on the B Bank of eprom/ram by using the CALL 000A instruction. See STEP 1 of CAT function, above.

2) The name of the file must be built up in the temp catalog area located in Bram. The filename itself must be 10d characters long and be stored in locations 3720 - 3729H. If the name is shorter than 10d characters long, it must be padded out with spaces. Location 372A must be set to indicate the type of file. File type codes used by SAFE for this purpose are: 00=Basic, 01=Numeric Array, 02=Character Array, 03=Bytes, 04=State, and 05=Variables. Particulars for each specific file type will be detailed below:
BASIC; (SAVE only) If execution is desired from a particular line number on loading, the line's number must be stored in locations 372D/372E before saving the file. If the program is NOT desired to autorun, 4000H should be stored here. The length of the Basic program should be stored in 372B/372C and the offset to the variables area in 372F/3730.
Arrays; The name of the array should be stored in AFST (3EF4) and CODF (3F1B) should be set to 00 if the particular array exists or non-zero if it does not. In general, access to DATA type array files from mc is very complex and will probably not be needed often.
BYTES; This type file (CODE or SCREEN$ from Basic) will probably be the most often needed type of file needing access from machine code. For a BYTES file, you must set 372A to 03 to indicate the type of file, store the start address of the save/load in 372D/372E and store the length or number of bytes to save/load in 372B/372C. If this is to be a LOAD, you must also set CODF (3F1B) to indicate whether or not it is desired to use the start address/number of bytes as was originally used when saved to disk instead. See CODF in the SAFE V2 disassembly DISK BRAM MAP AND SYSTEM VARIABLE TABLE for details as to the two bits used and what they represent. THESE BITS MUST BE SET PRIOR TO ANY BYTES LOAD FROM THIS DISK SYSTEM.
STATE; This type of file is similar to a BYTES file. To use a STATE save/load, set 372D/372E to 3E00 and 372B/372C to C200 so that save/load will be start at 3E00 in Bram and is C200 bytes long. The NMIF flag (3F0B) should reset to 00 to indicate that this save was not caused by a Non-maskable Interrupt.
VARIABLES; Store the start of the variables area in 372D/372E and their length in 372B/372C.

3) After the TCAT area is set up as required, the SAVE/LOAD flag at 378D must be set to 00h for a save, 80h for a merge, or FFh for a load.

4) Next step is to call the SAVE/LOAD/MERGE routine at location 1EAD. This routine will take the information you have provided in the TCAT area and perform the save/load/merge if possible as directed by the flag. If the routine can save or load the file without any errors, the carry flag will be set on return. If errors are encountered for any reason, chances are SAFE will try to return to Basic with an appropriate error message. From Basic, this can be fixed by using the ON ERR command, but from machine

code it is much harder to trap errors, as you must fiddle with the system variable ER SP (error stack pointer) to do so.

5) On completion of the save/load/merge, you must turn back off the B Bank and re-enable interupts. See the final step on accessing the CAT function, above, for details on correctly doing this.

6) If ERASE is to be performed instead of a load/save/merge, you only need to build up the name and type in step 2, then call ERASE at location 1CB6. Of course, step 1 and step 5 must also be done for ERASE.

The following is how the TCAT area from 3720 - 3735H is used by JLO SAFE V2:

3720 - 3729 = File name
372A = File type
372B & 372C = File length in bytes
372D & 372E = Beginning address of code save/load or line # in Basic program.
372F & 3730 = Offset to vars, Basic prog. only (372F=array name if data save/load)
3731 = File beginning track # (filled by SAFE)
3732 = File beginning side # (filled by SAFE)
3733 = # of cylinders used by file (filled by SAFE)
3734 & 3735 = Address of main cat entry if match found & over-write in progress. (filled by SAFE)

When loaded from disk to the B Bank buffer, the catalog area is made up as:

2600 = Tracks per side this disk
2601 = Max # of sides this disk
2602 & 2603 = Max # of cylinders available for use on empty disk
2604 & 2605 = # of free cylinders available for storage
2606 = Next free cylinder track # available for use
2607 = Next free cylinder side # available for use
2608 & 2609 = Next available catalog entry location. Starts at location 2620 and continues up till 33E8 or greater is out of range (177 files)
260A - 260F = Open for future use
2610 - 261F = Disk name. 16 characters.
2620 & up = First catalog entry. Each entry into this catalog is 20d bytes in length and is in the same exact same format as the TCAT area is made up as detailed above. (3720-3733)

NOTES ON ERRORS AND OTHER MISCELLANEOUS DATA

Errors encountered by JLO SAFE Disk Basic in reading or writing to disk are not ALWAYS reported with an error report to you. An example of this would be a DISK I/O ERROR in LOADing an ABS state file from disk that would result in part of the computer's ram being corrupted by having part of the file loaded into memory correctly but another part incorrectly loaded for one reason or another. This type of error is reported to you by a short "TOOT" at each try of reading the file, and finally if success in not found after a reasonable number of trys, a long "TOOT" being sounded through your computer's internal speaker. In this particular instance, Basic itself would also be NEWed to remove the corrupted file from memory thus avoiding a crash. If a file type other than the ABS state type is being loaded and errors are encountered after several trys, the computer will not NEW but report the problem to you with the long "TOOT" and the DISK I/O ERROR message. You are then free to try again with another disk or whatever you desire. SAFE Disk Basic will report the error EVERY time by sounding short "TOOTs" to alert you, and actually stopping with the DISK I/O ERROR message if the file can still not be loaded after several tries. These short "TOOTs" should alert you that the drive is having difficulty reading the disk. If you have not heard SAFE making these "TOOTs" before and it suddenly starts, you should clean your drive's heads, rearrange the Board 'A' data cable, or try anything else you can think of that could be causing the problem (perhaps you just changed to a different brand of diskette and your drive doesn't like the new brand, etc.). Another possibility is that the drive is being driven with a faster head step rate than this drive can handle or the drive requires more head settling time than is allowed with the head speed selected. If errors occur often then the slower head step speeds should be tried via the LET /H=n command to see if the drive selected is possibly being driven too quickly. Your system should rarely have errors at all, and if errors are frequent you SHOULD investigate the situation. You will know how often errors occur by use of this system, so if errors occur more frequently at sometime in the future you will realize that something has changed or needs cleaning.

Errors encountered when trying to FORMAT a disk can sometimes be caused by test tracks written on the disk at the factory for test/grading purposes. To understand why these tracks could cause a problem in formatting, you must realize that there is no erase head in any floppy disk drive! Data is erased by rewriting new data directly over the old data. If test tracks are written at the factory, the edges of the new track written may still contain part of the data put down at the factory, possibly causing errors when the FORMAT command verifies itself because this old data interferes with the new data. This would be even more likely to occur if the drive's head is not in the exact same alignment as the equiptment used in testing the diskette at the factory. So

how do you solve this kind of problem if it occurs? Use a small but strong permanent magnet to completely erase the disk by slowly rubbing it over the outside of the disk on both sides. Use care not to allow the magnet to actually touch the disk surface when near the cut-out areas where the media is exposed. While this is not a sure-fire cure for every formatting problem encountered, it should at least be tried before tossing a disk assuming it is defective. The method might save you some "wasted" money spent on what appear to be defective disks.

CAUTION...The WD1770 disk controller chip (especially) and ALL chips used on the Oliger Disk boards "A" and "B" can be damaged by static electricity! The WD1770 is a VERY expensive chip and may be hard to find in your area. Static prevention measures should ALWAYS be used when handling these parts or the boards they are installed in. The BEST preventive measure is to ALWAYS remove your shoes if you must handle these parts, especially in dry, low humidity areas like a normal home in the winter. It is advised that shoes be removed ANYTIME you sit down at your computer if possible. I personally zapped a cable converter and a digital clock a while back by just touching the cases of these devices. It CAN happen to you too!

One last "note" concerning power up of the computer with the Oliger 2068 Disk I/F. Additional time for power on reset has been added by the addition of C2 on Disk Board "A". This additional time is required by the WD1770 controller chip itself to allow it to properly reset on power up. Because this "on reset" time is lengthened, the amount of time required for the computer to be OFF before switching back on again is also lengthened. A quick flip off then on may no longer be sufficient to reset the system. A definite off then back on may now be required. You can tell if your complete system has reset by the default drive's LED indicator lighting on power up. If this LED does not light on power up, you should immediately switch power back off to try again till it does. You will soon get the "feel" for the required time for a proper reset.

USING JLO SAFE WITH THE ZEBRA OS64 CARTRIDGE

JLO SAFE will indeed work on the TS2068 with Zebra OS64 cartridge installed, but the NMI SAVE option will only work on OS64 cartridge versions V1.72 & up. Al Hartman of Zebra tells me that customers who have a copy of OS64 prior to V1.72 can upgrade to V1.72 (or later version if current one is newer) by returning their old OS64 cartridge to Zebra Systems with $5.00 to cover handling. I believe this is a very kind offer by Zebra and personally don't see how it can be beaten. I also believe Zebra's programmer for this cartridge (I won't mention his name as he or Zebra may not desire it) did an excellent job in implementing 64 column Basic on the TS2068 via an LROS cartridge.

Getting back to SAFE and the OS64 cartridge, as I said the disk system does work well with this cartridge, but only with the slight limitation that ABS saved files can only be LOADed with the machine in the same state as they were originally saved. This has nothing at all to do with the disk or cartridge hardware, but is brought about solely because of the entire state of the computer ABS SAVE requires the hardware to be configured exactly as is was when the file was saved, because the data in memory will be exactly as it was when it was saved and so the computer will probably expect to find the hardware also in the same state. All other types of save/loads (Basic programs, code, data, or variables) can be saved or loaded with the machine in OS64 mode or regular mode. In summary, using this disk I/F w/the Zebra OS64 cartridge is just as straightforward as ever, with the exception that ABS STATE files be loaded in the same mode they were saved in.

One more point to consider; Because of the 64 column screen with OS64, and the fact that OS64's regular COPY command supports the OLIGER PRINTER PORT, COPY / is of little or no value with OS64.

JLO SAFE DISK BASIC V2.5

EXTENDED COMMAND LIST

NOTE: In commands that use alphabetic characters (A-Z & a-z), lower case and upper case are treated as equal. For example, the command LET /D=1 is the same as LET /d=1. Commands that require a string (such as "FILENAME") will treat upper & lower case characters as totally different characters. Space and color/position control characters are ignored by SAFE Basic AFTER the "/" symbol, but nothing extra should be inserted BEFORE the "/" command identifier. All arguments shown as "n" can be either a number, variable, or expression, and if fractional will be rounded to closest integer. All SAVE/LOAD commands listed that require a "FILENAME" are allowed 10 characters for this name. Longer names will be truncated to 10 characters. Any character is allowed to be used in a "FILENAME", with the exception of CHR$ 128 which should not be the first character of any filename.

LET /S=n ;Sets maximum number of sides per drive to 1 or 2 for single or double sided drives. Default value = 2. Argument "n" must be in range 1-2 else INTEGER OUT OF RANGE. It is most important this be set correctly before a FORMAT command as this information is stored on the disk and will be used in the future for this disk only. EG: LET /S=1 LET /S=2+z/2 LET /s=X-2

LET /T=n ;Sets maximum number of tracks supported by your drive to "n". Default value = 40. Argument "n" must be in range 2-255 else INTEGER OUT OF RANGE. Again, this is stored on the disk

during FORMAT so be sure this is preset correctly before using the FORMAT command. EG: LET /T=80 LET /t=40 LET /t=tracks LET /T=x+2-y

LET /D=n ;Sets current drive to be drive "n". Default drive = drive 0. Argument "n" must be in range 0-3 else INVALID DRIVE # error. EG: LET /D=1 LET /d=0 LET /D=X+1

LET /H=n ;Sets head step rate to that required by your particular drive. Step rates are: 0=6ms (default), 1=12ms, 2=20ms, & 3=30ms. Argument "n" must be in range 0-3 else INTEGER OUT OF RANGE. IBM compatible drives can handle the default head step rate of 6ms, and should use this for fastest SAVEs & LOADs. Older drives may require a slower setting. Check your drive manual's specification page to find your drive's fastest step rate. EG: LET /H=3 LET /h=1 LET /h=step

LET /P=O and LET /P=T ;This non-disk related command selects the Oliger or Aerco Printer I/F w/o ANY disk/cassette LOADs, or reselects (LET /P=T) the Timex 2040 printer. This Extended Basic command allows quick and easy back & forth use of both the Oliger 2068 Printer I/F and the Timex 2040 printer. If your particular Oliger Port printer requires Line Feeds after Carriage Returns, then you must POKE 23324,10 before LPRINTing or LLISTing after the LET /P=O command is used. Also this driver can be set to hard format the lines printed to a maximum number of characters per line. Set this by POKing location 23323 the maximum number of characters desired to be printed per line. Default setting is 255. EG: LET /P=o: LPRINT "This line will be printed through the Oliger 2068 Parallel Printer Port": LET /p=t: LPRINT "This line has been redirected to go to the Timex 2040 printer". BEWARE: The regular Basic COPY command will wipe out the use of this command. If COPY is used, follow its use with another LET /P=O command BEFORE using the LPRINT command to your Oliger 2068 Printer Port. Use of COPY / will cause the Oliger Port to be re-selected automatically so this does not apply to use of COPY /.

LET /P=O/B and LET /P=T/B ;This is an optional way of entering the previous command, but with the added extension "/B" where "B" specifies the current COPY / protocol (see COPY / command and NMI COPY, later in this list). The letter used for "B" in the command will be "A" for ASCII character copy, "O" for OKIDATA copy, "L" for OLIVETTI PR2300 copy, "G" for GEMINI (and many EPSON) copy, and "B" for GORILLA BANANA copy. EG: LET /P=O/O LET /p=o/g LET /p=t/a

FORMAT /"DISKNAME" ;This command formats a blank disk in the current drive. Uses current maximum tracks and sides as set via the LET /S=n and LET /T=n commands, and these preset values will be the current values on all future accesses to this disk. Use care with this command as it will TOTALLY ERASE all files on a previously used disk. The maximum disk name length is 16

characters and if longer name will be truncated. EG: FORMAT /"Spectrum games" FORMAT /"2068 utilities" FORMAT /D$

SAVE /"FILENAME" or SAVE /"FILENAME" LINE n ;Saves current Basic program w/variables to disk. If LINE is used, will autostart on reLOAD at specified line #. Operation and Syntax of this command is identical to its cassette tape counterpart w/the "/" telling the system to use the disk instead of tape. If a file already exists with this name and it is also a Basic file, system will TOOT and report this, and allow you 5 seconds to abort the SAVE via use of the shift BREAK key. If not aborted in 5 seconds (or if ENTER is pressed to cancel 5 second wait) system will overwrite old file w/new file if there is enough space allotted to the old file to physically contain the new file. If there isn't, a FILE TOO LARGE error will occur with old file intact. If there is not enough room left on the disk to hold the program, a DISK FULL error will occur. EG: SAVE /"Letter" SAVE /"TREK" SAVE /x$ SAVE /"program" LINE 10

LOAD /"FILENAME" ;Compliment of SAVE /"FILENAME", above. Works exactly like its cassette tape counterpart using the disk drive instead of the cassette recorder. If system cannot find the filename specified, a FILE NOT FOUND error will occur. EG: LOAD /"Letter" LOAD /"TREK" LOAD /x$ LOAD /"program"

SAVE /0 ;This is a special save intended for a Basic user written "menu" program that will display current files on the disk and accept INPUT from the user to select and LOAD the file desired. Only one of these special SAVEs are allowed per disk and its length cannot exceed aprox. 1.5K else FILE TOO LARGE error. SAVE is physically located on the disk at side 0, track 0, sectors 1-3. Will sound a TOOT and return with DISK I/O ERROR if any errors are encountered in SAVing. EG: SAVE /0 SAVE /1-1

LOAD /0 or LOAD ;This command (simply <LOAD><ENTER>) is the easy way of LOADing in the Basic menu program SAVEd via SAVE /0, above. It is intended that, upon power up, a 2 key-press LOAD ENTER command will quickly and easily LOAD in a menu program which displays the contents of the disk. A single keypress from this menu program will then LOAD in the actual program desired. If no file 0 is on the current disk, system will return with FILE NOT FOUND error. If a disk read error is detected, this command will return with a DISK I/O ERROR. If ramtop is set so low that room is not available for this short program then a return will be made with an OUT OF MEMORY error. The real key to using this disk I/F is to set up an informative menu that displays the contents of the disk, what system it requires (2068 or Spectrum) and possibly what type of disk this is (Spectrum games, Mscript files, 2068 utilities, etc.). EG: LOAD LOAD /0

SAVE /"FILENAME" CODE n,m ;This command works exactly like its cassette based counterpart in both its Syntax and execution. INT

OUT OF RANGE error will occur if SAVE will wrap around past 65535 or if SAVE is from the very lowest part of memory because of the hardware design. If exact filename of same type already exists in the disk catalog, command will TOOT to inform you of this and allow 5 seconds to BREAK as in SAVE /"FILENAME", above, but will not print "FILE EXISTS!" message because save may be from this area of memory so it is desireable not to disturb the contents of the screen. EG: SAVE /"Ramcode" CODE 32768,8192 SAVE /"routine" CODE 40000,50 SAVE /"text" CODE x,m

LOAD /"FILENAME" CODE n,m ;This command is the complement of SAVE /"FILENAME" CODE n,m above. It is identical in Syntax & execution to its cassette based counterpart. If ",m" is left off, command will load as many bytes as were originally saved. If "n,m" is entirely left off, command will LOAD to the address originally saved from the number of bytes originally saved, just like the cassette based command does. EG: LOAD /"Ramcode" CODE 30000,8090 LOAD /"routine" CODE 32000 LOAD /"text" CODE

**SAVE /"FILENAME" SCREEN$** ;Actually a SAVE CODE using the memory area occupied by the bit-mapped video screen. Identical in Syntax & execution to its cassette based counterpart. EG: SAVE /"screen" SCREEN$ SAVE /a$ SCREEN$

**LOAD /"FILENAME" SCREEN$** ;Complement of SAVE /"FILENAME" SCREEN$, above. Identical in Syntax & execution to its cassette based counterpart. EG: LOAD /"screen" SCREEN$ LOAD /a$ SCREEN$

SAVE /"FILENAME" DATA X() ;Saves numeric array X to disk. Identical in Syntax & execution to its cassette based counterpart. EG: SAVE /"totals" DATA a() SAVE /b$ DATA e()

LOAD /"FILENAME" DATA X() ;Compliment of SAVE /"FILENAME" DATA X(), above. Identical in Syntax & execution to its cassette based counterpart. EG: LOAD /"totals" DATA a() LOAD /"b$ DATA e()

SAVE /"FILENAME" DATA X$() ;Saves character array X$ to disk. Identical in Syntax & execution to its cassette based counterpart. EG: SAVE /"text" DATA t$() SAVE /"messages" DATA M$()

LOAD /"FILENAME" DATA X$() ;Compliment of SAVE /"FILENAME" DATA X$(), above. Identical in function & execution to its cassette based counterpart. EG: LOAD /"text" DATA t$() LOAD /"messages" DATA M$()

SAVE /"FILENAME" VAL ;This is a new JLO SAFE V2 SAVE. Command will save all variables to disk from the current Basic program, but not the program itself. EG: SAVE /"VRBLS" VAL SAVE /a$ VAL

LOAD /"FILENAME" VAL ;Compliment of SAVE /"FILENAME" VAL, above. Variables presently within current program will NOT be cleared automatically by this command. If you desire the present

variables deleted before the new variables are loaded, simply use the CLEAR command before loading the new variables. If a new variable is loaded whose name is already a variable within the program, BOTH will physically exist in memory but Basic will only find the old variable when references are made to the shared variable name. EG: LOAD /"variables" VAL LOAD /a$ VAL

SAVE /"FILENAME" ABS ;This new JLO SAFE V2 SAVE will save the entire "STATE" of he computer to disk. This will include ALL of the regular 2068 ram, so Basic program, variables, machine code, video screen, etc. will ALL be saved to disk in one shot. This command replaces the JLO SAFE V1 SAVE /n command doing exactly the same thing the older command did. EG: SAVE /"MS & text" ABS SAVE /A$ ABS

LOAD /"FILENAME" ABS ;Compliment of SAVE /"FILENAME" ABS, above. When file is finished loading, the computer will be in the EXACT STATE it was when the file was originally saved (IE: The computer will think it just saved the file!). This command is also used to load files saved with the SAFE V2 NMI pushbutton SAVE feature. EG: LOAD /"MS & text" ABS LOAD /A$ ABS LOAD /"1" ABS

LOAD /n ;This command is a carry-over from JLO SAFE V1, which allows SAFE V1 files to be loaded with SAFE V2. You can load SAFE V1 files using this command, but you can only re-save the file on a SAFE V2 formatted disk with an initialized catalog. DO NOT ATTEMPT TO SAVE ONTO A SAFE V1 FORMATTED DISK USING JLO SAFE V2. EG: LOAD /1 LOAD /file

SAVE //"FILENAME"type ;This variation of the SAVE command using two "/" instead of only one tells SAFE to save a file without the "FILE EXISTS! 5 SECONDS TO ABORT" message and its "TOOT" w/delay. The idea is to use this variation of the SAVE command when you already know or expect the filename used to be present on the disk and wish to overwrite it without the delay and warnings that would occur in doing so using the regular SAVE / command. This is most often needed on a program such as a BBS that is running unattended and constantly updating files. Situations like this do not need this warning or the delay it causes, so this command gives the programmer a way around them. EG: SAVE //"userbase" DATA U$() SAVE //"EMAIL" DATA E$() SAVE //"newtext" CODE 32768,10000

MERGE /"FILENAME" ;This command is a simplified but VERY fast Basic program merge. Using this command instead of LOAD will append a Basic program with its variables (if any are present) onto the end of the current Basic program, without erasing the current program or its variables. This command will ONLY append the merged program and NOT insert lines or delete current lines or variables. The merged program should be written with high line numbers with this restriction born in mind. This command can be used within a running program, along with use of the regular 2068

DELETE command, to make far larger Basic programs than are normally possible by merging subroutines in as needed, using them, deleting them, merging more as nec., etc.. Room is dynamically opened for the merged code and loaded directly in so speed is VERY fast even for very long subroutines. Variables merged are stored at the beginning of the variables area, so beware that if a new variable is merged with a Basic program that has the same name as a variable used by the main program, Basic wll find and use the merged variable rather than the main variable. The main program variable will still be in memory, but Basic will not be able to access it. Beware also that if merged code has lower line numbers than the main Basic program, Basic is very likely to become confused and stop with an error. The limitation that the merged code have higher line numbers than the main program's code was considered a good trade for the very much increased speed of loading the new code in within a running program. The size of programs merged using a disk system in this way tends to increase making the time required for Basic to actually insert & delete lines as is normally done with a cassette merge simply too long to be really pratical. EG: MERGE /"9000 gosub" MERGE /"menue#1" MERGE /G$

CAT or CAT / ;This command will LOAD the catalog from the current drive and display information detailing the contents of the disk on the screen. "CYLS" referred to in this disk catalog are blocks of disk space used by the SAFE V2 DOS in allocating disk space to particular files. Each cylinder represents 5K (5120) bytes of physical disk space. Because space is allocated in 5K cylinders, a file 2 bytes long uses the same amount of disk space as a file 5K bytes long. Although space is reserved in 5K cylinders, actual SAVing and LOADing works only on the exact amount of bytes needed. Thus, if you entered the command SAVE /"byte" CODE 30000,1 , you would only really SAVE 1 byte to disk, but 5K bytes of disk storage space would be reserved for the file "byte". You could then later SAVE with SAVE /"byte" CODE 29000,4000 as an update to this file and the new data would still fit in the space reserved for the CODE file named "byte". The CATalog command will list both the number of cylinders allocated to the file and the true length of that file. An ABS SAVE will be listed as being a STATE type file and a VAL SAVE will be listed as being a VRBLS type file. A Basic program w/variables will be called a BASIC file, a CODE or SCREEN$ SAVE as a BYTES file, a DATA a() SAVE as a N ARR file, and a DATA a$() SAVE as a C ARR file. If you have a long catalog resident on your disk and would like the catalog printed through your Oliger Printer Port instead of the screen, then do the following: 1)Select Oliger Port with the LET /P=0 command. 2)Redirect stream #2 to the Oliger Printer Port by using the command OPEN#2,"P". 3)Send the CATalog to the printer with the CAT command. 4)Redirect stream #2 to the screen by using the command CLOSE#2. EG: CAT CAT /

NMI SAVE ;This function is not actually an Extended Basic

command, but a hardware function of Oliger Disk Board "B". The press of this button allows the SAFE eprom to save the entire state of the machine to disk at ANY time desired, and several other functions. This function is very desirable because it allows use of the Oliger Disk System with 99.9% of software available for the 2068 or Spectrum Emulating 2068. The NMI saved file is loaded normally when desired using the LOAD /"FILENAME" ABS command. NO MORE PATCHING AND TRYING TO BREAK INTO LOAD & RUN PROGRAMS! A simple pushbutton press forces the current program running to stop and the computer will "toot" to acknowledge that it is awaiting a key press. After a press of keys 1 through 0 is detected the current state of the machine is stored on disk as an ABS STATE file with a filename of "0" - "9" as directed through the keypress. If the SAVE is not successful, the computer will play a long "toot" to inform you and return to the current program. You can then change to another disk or whatever and try again as needed. If you change your mind after pressing the SAVE button, a press of the ENTER key will return control of the machine to the program that was running. Thus this function will double as a PAUSE ANYTIME function.

NMI SCREEN COPY: NMI SAVE button press and then copy key press (Z key) will cause the COPY / routine to take control and dump the current screen to the Oliger 2068 Printer Port (or Aerco I/F). Protocol used will be as last set via the LET /P=0/B command or an ASCII character copy if this command has not been used since power up, as the ASCII copy is the default copy / routine supported. NOTE: Because the ASCII copy routine makes heavy use of the stack, it can very easily crash some stack dependent MC programs. The hi-resolution copy routines do not use the stack nearly so heavily, so they are unlikely to cause problems of this sort. If in doubt, SAVE to disk before using the ASCII NMI COPY just to be sure.

NMI SCREEN$ SAVE: A press of keys Q, W, E, R, or T causes the curent screen to be sent to disk as a SCREEN$ BYTES file whose filename will be "A" through "E" respectively. Now you can save the screen anytime to reload later to print with whatever you like or modify with an art type program!

NMI BUTTON BREAK: A press of the "C" key after the NMI SAVE pushbutton has been pressed will cause an attempt to return to Basic. This routine is not an exhaustive attempt at recovering the computer from protected software, but a "last resort" before pulling the plug on a run-away program. It MIGHT return the computer to Basic or could cause a complete & non-recoverable crash. Use it only as a last resort or if you don't care if the computer does crash. This function WILL recover control of the computer from a never ending mc loop entered from Basic with the USR function.

RESET: Holding the "N" (for NEW) key and a short tap of the NMI SAVE button will cause a system reset. This function is like adding a reset switch without physically adding one.

MOVE / ;This command will make a copy of the entire disk in the

current drive onto the disk in the next logical drive. If the current drive is drive 3, then the next logical drive is considered drive 0. This command uses the max. track and side setting as set via the let commands, so beware. This command is only for making back-ups of the entire disk using two identical drives. If your drives are not identical, you must use the command below (which can handle this) and move the files one at a time. EG: MOVE /

MOVE /"FILENAME"type (or) MOVE /"FILENAME" type TO n ;This command will move a copy of the file "FILENAME" with a filetype of "type" (Basic default if ommitted) from the current drive to drive "n" as specified in the command. If a destination drive is not specified by leaving off "TO n", command will default to move the file to the next logical drive # as is normally done with MOVE /, above. If file cannot be located on current drive, will return with error W File not found. If disk in drive n does not have enough free file space left to hold a copy of the file, will return with error U Disk full. If drive specified as "n" is out of the range 0-3, will return with error W Invalid drive #. This command does NOT check for the filename already being used on the target drive, so multiple copies of the same file/filename can be moved to a single disk. SAFE will ALWAYS use the first correct filename/type found when loading, saving, renaming, etc., so files with the same filename and type can be considered protected when listed via CAT lower in the listing than another file with the same name. After changing the topmost filename via RESTORE /"FN" TO "NFN", the then topmost listed file can be accessed by SAFE. EG: MOVE /"program" MOVE /"text"CODE TO 1 MOVE /A$ MOVE A$ ABS MOVE /"variables"VAL TO 3

RESTORE /"OLDFNAME"type TO "NEWFNAME" ;This command will re-name the present file "OLDFNAME" the newly selected filname "NEWFNAME". If the file is anything other than a Basic program file, a type code of CODE, SCREEN$, DATA, DATA $, VAL, or ABS must be included after the old filename. For correct syntax, use only the keyword DATA for a numeric array file or DATA followed by $ for a character array file; Do not use the actual array letter as used with the SAVE/LOAD commands. The old file name must match exactly the name as it actually appears within the catalog. If SAFE cannot match the old file name and type, a FILE NOT FOUND error will occur. EG: RESTORE /"Space inv" TO "Space Inva" RESTORE /"MScpt bas" TO "Mscript Ba" RESTORE /"old data" DATA TO "data V2" RESTORE /X$ ABS TO N$ RESTORE /"1" ABS TO "NMI SGAME" (use of this command allows greater than 10 NMI SAVE files to be stored directly on a single disk by renaming the files after saving).

RESTORE /"NEW DISK NAME" ;This command will re-name the disk in the current drive to become "NEW DISK NAME". Maximum disk name length is 16 characters. If longer name will be truncated. EG: RESTORE /"Disk number 34" RESTORE /D$

RESTORE /S ;This command will reinitialize SAFE and restore all defaults to their power-on configurations. This command can be used at any time to insure power-on defaults. This command can also be used before a SAFE command or error is encountered if the computer is powered up by accident with the Board B enable/disable switch off. CAUTION: Using any SAFE command or causing any error with SAFE switched in (via use of enable switch and/or SA/OK switch) but not initialized via use of this command or by having SAFE active on power-up, is VERY likely to cause a crash! EG: RESTORE /S RESTORE /s

ERASE /"FILENAME"type ;This command will erase the file "FILENAME" on the disk in the current drive. Use this command with care and only if you are SURE you really want the file erased. Back up your disk before using this command "just in case"! EG: ERASE /"obsolete" CODE ERASE /"junk file" ERASE /"too little" VAL

VERIFY /"FILENAME"type ;This command will check the file "FILENAME" on the current drive and make the WD1770 controller chip perform CRC error checks on all the data contained within the file's allotted space. Because an actual compare to memory is NOT made, this command can be used at any time on any file desired to insure it IS stored correctly for whatever purpose. SAFE V2 automatically verifies all files saved without using this command, but this command is installed as a "second check" to be absolutely sure a very important file is ok. EG: VERIFY /"Sky Attack" VERIFY /"Bobby"ABS VERIFY /B$ CODE

COPY / ;This command will copy the current display file through the Oliger 2068 Printer Port. The protocol used in copying the screen will be as previously set with the LET /P=O/B command. If the TS2040 printer is also connected and turned on, it will also make a copy before the big printer copy is performed. EG: COPY /

FOR /n TO m or FOR /m ;This new command sets the start and range of a new fast integer only FOR-NEXT. n and m must be in the range of 0-65535. If m is set to 65535, the machine will loop forever. A STEP cannot be used (always +1) and no nested loops are allowed (only one SAFE LOOP at a time, but they CAN be nested with the slower Basic FOR/NEXT loops). The control variable name used is as set via use of a LET X=n at the very beginning of a Basic program or directly after a CLEAR command (IE: The simple (single letter) variable MUST be stored at the very beginning of the variables area). If the FOR /n TO m command is executed and a LET command has not been executed at the very start of the program, the computer will return with a 'FOR/ w/o LET' error. If

the variable declared in the LET statement becomes negative by Basic manipulation, the computer will return with a ´neg FOR/ variable´ error on encountering the next ´NEXT´ statement. If the variable declared in the LET statement becomes a floating point variable by Basic manipulation, the computer will return with a ´fp FOR/ variable´ error on encountering the next ´NEXT´ statement. This new FOR/NEXT loop is desirable to use in place of a regular Basic FOR/NEXT loop where speed is important. The execution time of the new SAFE FOR/NEXT loop is 9 to 50 (or MORE) times faster than a regular FOR/NEXT loop, depending on its placement in the Basic program. (IE: A regular Basic loop gets slower the further it is within the program while this new looping structure is position independant speed-wise). If the ´n TO´ is left off the command, the loop will start by setting the control variable to 1 (IE: FOR /1 TO 255 is identical to FOR /255 in operation) EG: FOR /2 TO 220 FOR /0 TO 100 FOR /1000 FOR /start TO end FOR /m-2 TO m+5

NEXT ;This is the looping statement for the new FOR-NEXT loop described above. No arguments are allowed. Will cause control of Basic program to be passed to the next statement after the FOR / command if the control variable has not exceeded its limit as set in the FOR / command. EG: NEXT

(The new fast SAFE loop will not work on a Basic AROS cartridge unless the FOR / and NEXT are both on the same Basic line.

One more "FUNCTION" has also been implemented in JLO SAFE V2. If one of the keys 1, 2, or 3 is held down on power-up, this will cause the power on default drive to be drive 1, 2, or 3 instead of the normal default active drive 0. This feature makes using the LET /D=n command not now required when powering up and intending to use other than the default drive zero.

NOTICE: THE NMI SAVE ANYTIME FEATURE OF JLO SAFE IS INTENDED AS THE MOST SATISFACTORY METHOD OF TRANSFERRING PROGRAMS USED ON THE TIMEX/SINCLAIR COMPUTERS TO THIS DISK SYSTEM, AND NOT AS A MEANS OF PIRATING COPYRIGHT PROTECTED SOFTWARE. USE OF JLO SAFE FOR OTHER THAN LAWFUL DISK BACK-UP OF COPYRIGHTED SOFTWARE IS ILLEGAL AND CERTAINLY NOT ENDORSED IN ANY MANNER BY JOHN OLIGER CO.

JLO SAFE V2.5 COMMAND LIST SUMMARY

```
LET /S=n                                  LET /T=n
LET /D=n                                  LET /H=n
LET /P=0 & LET /P=T                       LET /P=0/B & LET /P=T/B
FORMAT /"DISKNAME"                        SAVE /"FILENAME"
SAVE //"FILENAME" ;Allows saving w/o warning message, all types
SAVE /"FILENAME" LINE n                   LOAD /"FILENAME"
SAVE /0                                   LOAD or LOAD /0
SAVE /"FILENAME" CODE n,m                 LOAD /"FILENAME" CODE n,m
LOAD /"FILENAME" CODE n                   LOAD /"FILENAME" CODE
SAVE /"FILENAME" SCREEN$                  LOAD /"FILENAME" SCREEN$
SAVE /"FILENAME" DATA X()                 LOAD /"FILENAME" DATA X()
SAVE /"FILENAME" DATA X$()                LOAD /"FILENAME" DATA X$()
SAVE /"FILENAME" VAL                      LOAD /"FILENAME" VAL
SAVE /"FILENAME" ABS                      LOAD /"FILENAME" ABS
LOAD /n                                   MOVE /
MOVE /"FILENAME"type TO n                 MOVE /"FILENAME"type
COPY /                                    RESTORE /S
RESTORE /"OLDNAME"type TO "NEWFNAME"      RESTORE /"NEW DISK NAME"
VERIFY /"FILENAME"type                    FOR /n
FOR /n TO m                               NEXT
MERGE /"FILENAME"type                     CAT or CAT /
ERASE /"FILENAME"type
```

NMI PUSHBUTTON FUNCTIONS

ABS STATE SAVE via keys 1 through 0
SCREEN$ SAVE via keys Q through T
COPY SCREEN to Oliger Printer Port via key Z
BREAK to Basic w/the OK eport via key C
Return to interrupted program (pause) via the ENTER key

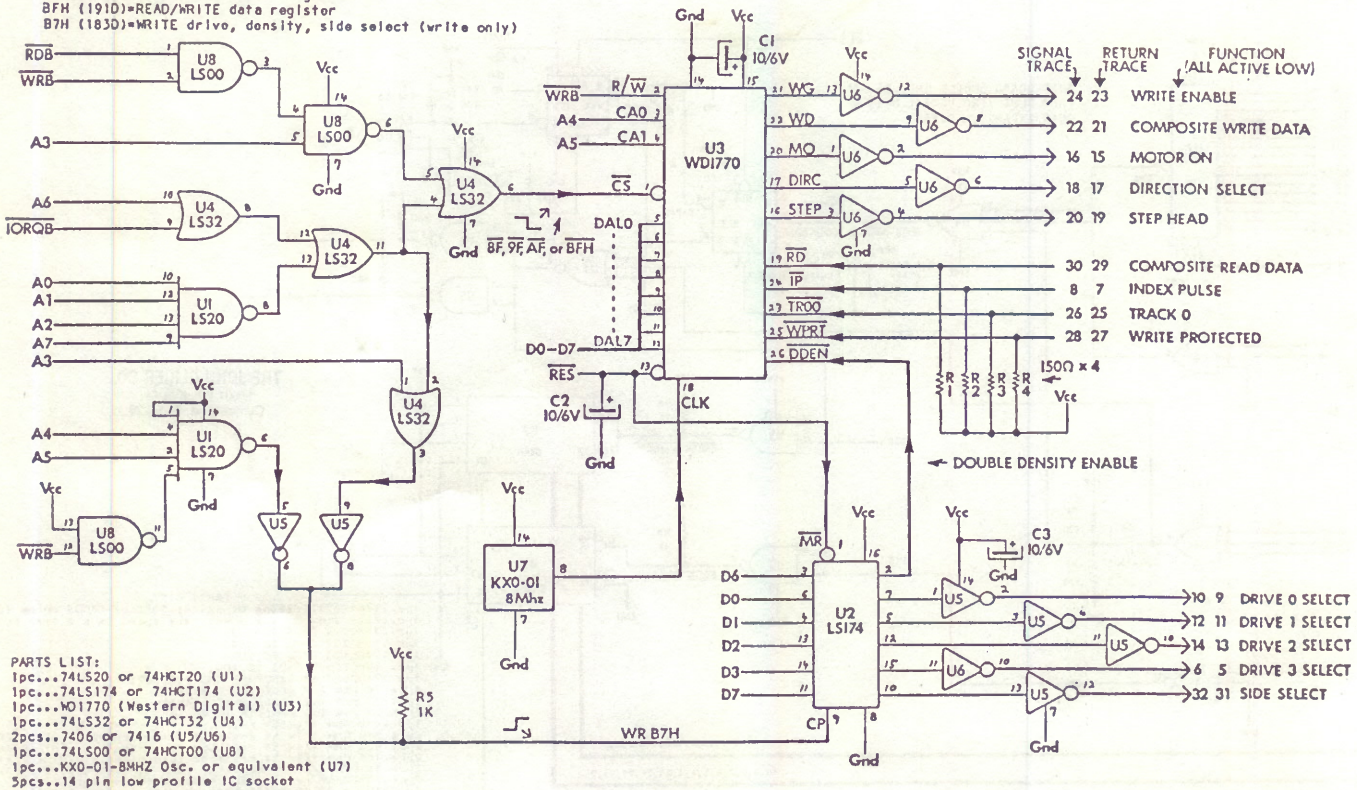
Holding key "N" (for NEW) and a tap of the NMI button causes a system reset.

Holding keys 1, 2, or 3 on reset or power-up causes the default drive to become drive 1, 2, or 3 instead of drive 0.

Blank Page

OLIGER 2068 DISC INTERFACE BOARD "A"

PORT ASSIGNMENTS: (All are fully decoded)
8FH (143D)=READ status register/WRITE command register
9FH (159D)=READ/WRITE track register
AFH (175D)=READ/WRITE sector register
BFH (191D)=READ/WRITE data register
B7H (183D)=WRITE drive, density, side select (write only)

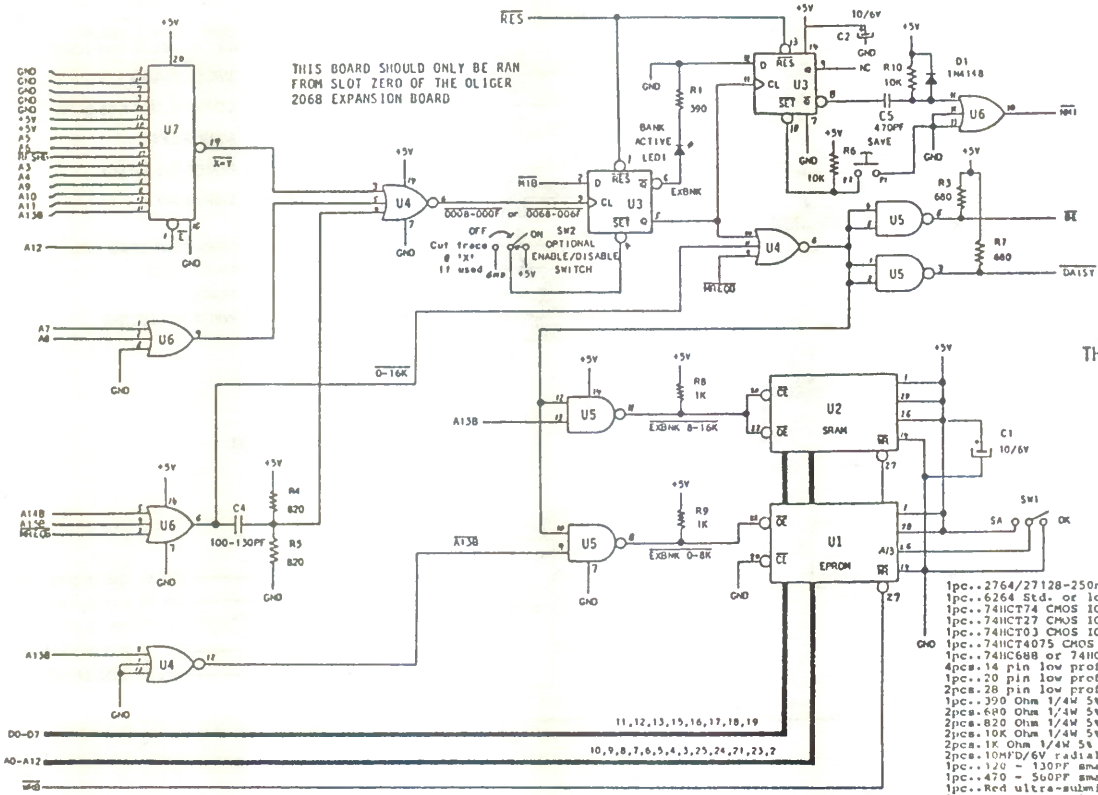


PARTS LIST:
1pc...74LS20 or 74HCT20 (U1)
1pc...74LS174 or 74HCT174 (U2)
1pc...WD1770 (Western Digital) (U3)
1pc...74LS32 or 74HCT32 (U4)
2pcs..7406 or 7416 (U5/U6)
1pc...74LS00 or 74HCT00 (U8)
1pc...KX0-01-8MHZ Osc. or equivalent (U7)
5pcs..14 pin low profile IC socket
1pc...16 pin low profile IC socket
1pc...28 pin low profile IC socket
4pcs..150 Ohm 1/4W 5% resistor (brn-grn-brn-gold)
1pc...1K Ohm 1/4W 5% resistor (brn-blk-red-gold)
3pcs..10MFD/6Volt radial lead tantalum capacitor

OLIGER 2068 DISK INTERFACE BOARD 'B'

FIRMWARE/RAM/BANKING SUPPORT

rev. 'A'



THE JOHN OLIGER CO.
11601 Whidbey Dr.
Cumberland, IN 46229

PARTS LIST

1pc..2764/27128-250ns or faster SAFE or OKIDOS eprom (D1)
1pc..6264 Std. or low power, any speed 8K x 8 SRAM (U2)
1pc..74HCT74 CMOS IC (U3)
1pc..74HCT27 CMOS IC (U4)
1pc..74HCT03 CMOS IC (U5)
1pc..74HCT4075 CMOS IC (U6)
1pc..74HC688 or 74HCT688 CMOS IC (U7)
4pcs.14 pin low profile IC socket
1pc..20 pin low profile IC socket
2pcs.28 pin low profile IC socket
1pc..390 Ohm 1/4W 5% resistor (org-wht-brn-gld) (R1)
2pcs.680 Ohm 1/4W 5% resistor (blu-grey-brn-gld) (R3/R7)
2pcs.820 Ohm 1/4W 5% resistor (grey-red-brn-gld) (R4/R5)
2pcs.10K Ohm 1/4W 5% resistor (brn-blk-org-gld) (R6/R10)
2pcs.1K Ohm 1/4W 5% resistor (brn-blk-red-gld) (R8/R9)
2pcs.10MFD/6V radial lead tantalum Capacitor (C1/C2)
1pc..120 - 130PF small mono or disc capacitor (C4)
1pc..470 - 560PF small mono or disc capacitor (C5)
1pc..Red ultra-subminiature LED (LED1)
1pc..1N4140 or 1N914 switching diode (D1)
1pc..SPST ultra-subminiature pc mount slide switch (SW1)
1pc..Optional SPST ultra-subminiature pc mount slide switch (SW2)
1pc..pc mnt norm off momentary pushbutton switch (SW3)

MACHINE CODE USED:

```
CDØAØØ   MVCT CALL ØØØA
2A4B5C        LD HL,(vars)
11ØEØØ        LD DE,ØØØE
19            ADD HL,DE
EB            EX DE,HL
212Ø26        LD HL,CTFL
Ø1D4ØD        LD BC,ØDD4
EDBØ          LDIR
211Ø26        LD HL,DNAM
ØE1Ø          LD C,1Ø
EDBØ          LDIR
111ØØØ        LD DE,ØØ14
3E8Ø          LD A,8Ø
BE       FLLP CP (HL)
28Ø4          JR Z,RETI
Ø3            INC BC
19            ADD HL,DE
3ØF9          JR NC,FLLP
DF       RETI RST 18H
```

mc BYTES LISTING:

HEX		DEC	CHR$
CD		205	STEP
ØA		1Ø	
ØØ		Ø	
2A		42	*
4B	vars	75	K
5C		92	\
11		17	
ØE		14	
ØØ		Ø	
19		25	
EB		235	FOR
21		33	!
2Ø	CTFL	32	
26		38	&
Ø1		1	
D4		212	CLOSE #
ØD		13	
ED		237	GO SUB
BØ		176	VAL
21		33	!
1Ø	DNAM	16	
26		38	&
ØE		14	
1Ø		16	
ED		237	GO SUB
BØ		176	VAL
11		17	
14		2Ø	
ØØ		Ø	
3E		62	>
8Ø		128	
BE		19Ø	PEEK
28		4Ø	(
Ø4		4	
Ø3		3	
19		25	
3Ø		48	0
F9		249	RANDOMI
DF		223	OUT

39 BYTES LONG
(QS)